RELIABILITY OPTIMIZATION
OF COMMUNICATION NETWORK TOPOLOGY DESIGN
USING IMPROVED ANT COLONY ALGORITHM

KANYAPAT WATCHARASITTHIWAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE DEGREE OF DOCTOR OF ENGINEERING
IN ELECTRICAL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2009

KMITL-2009-EN-D-018-151

หัวข้อวิทยานิพนธ์	การหาค่าความเหมาะสมของความน่าเชื่อถือได้ของการออกแบบโทโปโลยีโครงข่ายการสื่อสาร โดยใช้การหาค่าที่เหมาะสมแบบวิธีฝูงมดที่ปรับปรุงแล้ว
นักศึกษา	นางสาวกัณญพัชร์ วัชรสิทธิวัฒน์
รหัสนักศึกษา	47060058
ปริญญา	วิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2552
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.ปราโมทย์ วาดเขียน

บทคัดย่อ

ปัญหาการออกแบบโครงข่ายนั้นเป็นที่ทราบกันดีว่าเป็นปัญหาแบบ NP-Hard ซึ่งรวมทั้งการคัดเลือกกลุ่มของการเชื่อมต่อหรือโทโปโลยีของโครงข่าย เพื่อลดค่าใช้จ่ายหรือต้นทุนของโครงข่าย โดยอยู่ภายใต้เงื่อนไขของค่าความน่าเชื่อถือได้ที่กำหนดให้ จากปัญหาดังกล่าว วิทยานิพนธ์ฉบับนี้จึงเสนอระเบียบวิธีการที่มีประสิทธิภาพแบบใหม่ ซึ่งอยู่บนพื้นฐานของการหาค่าที่เหมาะสมแบบวิธีฝูงมดดั้งเดิม (Conventional Ant Colony Optimization – ACO) เพื่อแก้ไขปัญหาการออกแบบโครงข่ายสื่อสาร เมื่อพิจารณาทั้งทางด้านค่าใช้จ่ายและค่าความน่าเชื่อถือได้ของโครงข่ายนั้น

วิธีการที่นำเสนอในวิทยานิพนธ์ฉบับนี้ชื่อว่า การหาค่าที่เหมาะสมแบบวิธีฝูงมดที่ปรับปรุงแล้ว (Improved Ant Colony Optimization - IACO) ซึ่งได้นำเสนอเทคนิคเพิ่มเติม 2 แบบ เพื่อปรับปรุงขบวนการค้นหา คือวิธีการค้นหาคำตอบรอบข้าง (Neighborhood Search) และขบวนการเริ่มต้นการค้นหาครั้งใหม่ (Re-initialization) จากนั้นได้นำการหาค่าที่เหมาะสมแบบวิธีฝูงมดที่ปรับปรุงแล้ว ไปประยุกต์ใช้กับโทโปโลยีของโครงข่ายที่แตกต่างกัน 3 แบบ เพื่อแสดงประสิทธิภาพของวิธีการที่นำเสนอ พร้อมทั้งเปรียบเทียบผลลัพธ์กับวิธีแบบดั้งเดิมอื่นๆ คือ วิธีพันธุกรรมยีน (Genetic Algorithm - GA) วิธีการค้นหาแบบตาบู (Tabu Search Algorithm - TSA) วิธีการหาค้นหาแบบกลุ่มอนุภาค (Particle Swarm Optimization - PSO) และวิธีฝูงมดดั้งเดิม ในแต่ละตัวอย่างได้ทำการทดสอบภายใต้เงื่อนไขหลายค่า ซึ่งผลลัพธ์จากการจำลองสถานการณ์แบบต่างๆ แสดงให้เห็นว่าวิธีการที่นำเสนอนั้นมีสมรรถนะเหนือกว่าวิธีการแบบดั้งเดิมอื่นๆ ทั้งในแง่ของคุณภาพและเวลาในการคำนวณหาคำตอบ

Thesis Title	Reliability Optimization of Communication Network Topology Design Using Improved Ant Colony Optimization
Student	Ms. Kanyapat Watcharasitthiwat
Student ID	47060058
Degree	Doctor of Engineering
Program	Electrical Engineering
Year	2009
Thesis Advisor	Assoc. Prof. Dr. Pramote Wardkein

ABSTRACT

Network design problem is a well-known NP-hard problem which involves the selection of a subset of possible links or a network topology in order to minimize the network cost subject to the reliability constraint. To overcome the problem, this thesis proposes a new efficiency algorithm based on the conventional ant colony optimization (ACO) to solve the communication network design when considering both economics and reliability.

The proposed method is called improved ant colony optimizations (IACO) which introduces two addition techniques in order to improve the search process, i.e. neighborhood search and re-initialization process. To show its efficiency, IACO is applied to test with three different topology network systems and its results are compared with those obtained results from the conventional approaches, i.e. Genetic Algorithm (GA), Tabu Search Algorithm (TSA), Particle Swarm Optimization (PSO) and ACO. Simulation results, obtained in these test problems with various constraints, show that the proposed approach is superior to the conventional algorithms both solution quality and computational time.

Acknowledgement

This thesis would not have been possible without helpful and invaluable advice of Associate Professor Dr. Paramote Wardkein, who is my thesis adviser, including Asst. Professor Dr. Jeerasuda Koseeyaporn, who is of great assistance in very informative academic articles I am very grateful to all of them for their kind and indefatigable support. including

I would like to thank all members of committee for their valuable comments and also acknowledge my gratitude to all faculty members of the Telecommunications Engineering Department, Faculty of Engineering, King Mongkut's Institute of Technology, Lardkrabang, who had taught me throughout the course of my study.

My thanks also go to the Graduate Studies and Faculty of Graduate Studies, Faculty of Engineering for their contributions to my pursuit of further knowledge in Engineering.

Last but not least, I am most indebted to my father, mother, family and friends, for their ceaseless morale support and kind assistances in everything, without which this thesis would not have been possible.

For any values and benefits to be incurred by this thesis, I sincerely dedicate them to all those to whom I owe a heavy debt.

Kanyapat Watcharasitthiwat

Table of Contents

Page

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

Figure Page

List of Figures (Continued)

List of Figures (Continued)

List of Tables

CHAPTER 1

Introduction

This thesis is mainly focused on a performance improvement of the Swarm Intelligent search method in order to solve the real-world optimization problems in telecommunication network operation and planning. Although many heuristic algorithms and swarm intelligent techniques such as Genetic Algorithm (GA), Tabu Search (TS), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have been successfully used to solve the telecommunication and computer network design problems, there are some weaknesses and obstacles which can be improved and developed to get better solution.

Nowadays, the personal microcomputer has high speed computation and low cost. This allows optimization techniques to improve performance of solving real world problem effectively and efficiently. Therefore, this thesis has proposed a new optimization technique based on Ant Colony Optimization in order to enhance the performance of the method. Furthermore, the additional mechanisms namely, neighborhood search and re-initialization help to improve the search process in terms of both solution quality and computational time.

1.1 Literature Review and Contribution of Problem

A communication network can be illustrated by set of nodes (or switches) and links (or arcs) where all nodes are connected by links. The typical communication network structure is composed of two levels. The first one is backbone network and the second one is local access network (LAN). The backbone network is dedicated for delivery of information from source to destination (end to end) using its switching nodes. The LAN network is a typically centralized system which allows users to access hosts or local servers. This paper is focused only on the backbone network design considered as a distributed network.

The advent of low-cost devices has led to explosive growth in communication networks. The topology network design is a part of network planning which finds a suitable topology in order to satisfy some constraints. One of the major advantages of the distributed network over the centralized system is its flexibility to improve the system reliability. The reliability of a system depends not only on the reliability of its nodes and links but also on the topology of a network. A completely connected network has the highest network reliability while the simple loop (ring) network has the lowest network reliability.

In the past, many researchers had studied network design considering the system reliability as a constraint or an objective. The reliable network design problems are stated as two-terminal network reliability and all-terminal network reliability (also known as overall reliability). The problem of optimum topology network design, which selects the links that either maximizes reliability or minimizes cost, can be formulated as a combinatorial problem.

This problem is a well-known NP-hard problem. The total number of possible solution is;

$$\text{Total number of possible solution} = k^{\frac{(|N|\times|N|-1)}{2}} \tag{1.1}$$

where k : Options of each link

N : Set of given nodes

For such a problem, the researchers have studied with an enumerative-based and heuristic method. Jan et, al. [1] developed an algorithm using the decomposition approach based on brand and bound to minimize link cost of a communication network subjected to a reliability constraint. Aggárwal et, al. [2] employed a greedy heuristic approach to maximize reliability given a cost constraint for networks with different reliability of links and nodes. Pierre et, al. [3] also used simulated annealing to find the optimal design for packet switching networks where delay and capacity were considered, but reliability was not. For the network design, Kumar et, al. [4] developed a genetic algorithm (GA) considering diameter, average distance and communication network reliability and then applied it to four test problems of up to nine nodes. Deeter and Smith [5] presented a GA approach for minimum-cost network design problem with alternative link reliabilities and all-terminal network reliability constraint. Furthermore, Glover et, al. [6] used the tabu search (TS) algorithm to choose topologies of networks when considering cost and capacity but not reliability. For other work of TS algorithm, Beltran and Skorin-Kapov [7] used TSA to design reliable networks for searching the least-cost spanning two-tree where the two-tree objective was a coarse surrogate for reliability.

Recently, an ant colony optimization (ACO) approach has been proposed [8, 9] and successfully applied to the combinatorial optimization problem, such as traveling salesman problem (TSP) [10, 11], quadratic assignment problem [12], vehicle routing problem [13] and job-shop scheduling problem. Nevertheless, the ACO method has not yet been applied to design the topology of communication networks subjected to a reliability constraint.

This thesis presents a new heuristic approach based on ACO to design the topology of communication networks while considering two-terminal reliability and all-terminal reliability formulated as cost minimizing given a reliability constraint. The new approach is called improved ant colony optimization (IACO), which introduces additional improvement procedures, i.e. neighborhood search and re-initialization in order to develop and improved the algorithm to be able to get better solution with less time.

1.2 Objective of Thesis

1.2.1 Study and develop the techniques of designing the optimal telecommunication network, considering the reliability as a constraint

1.2.2 Apply the swarm intelligent optimization to design a telecommunication network

1.2.3 Develop a new technique in order to improve the performance of telecommunication network design

1.3 Scope of Thesis

1.3.1 Apply successfully the Swarm Intelligent Optimization technique to the design of a telecommunication network in order to minimize investment cost, considering the reliability as a constraint

1.3.2 Consider the network design with two-terminal and all-terminal reliabilities

1.3.3 Test the performance of the Swarm Intelligent Optimization technique with different telecommunication networks

1.4 Outcome of Thesis

1.4.1 An improvement in the efficiency of solving the telecommunication and computer network design problems

1.4.2 An ability to apply these techniques to the practical telecommunication networks

1.4.3 An ability to apply these techniques to solving other applications

1.4.4 The obtained results are advantages to society

1.5 Outline of Thesis

The remainder of this thesis is organized as follows: Chapter 2 describes the problem formulation, assumptions of system and reliability calculation. Chapter 3 discusses the basic principle of optimization methods i.e. Genetic Algorithm (GA), Tabu Search Algorithm (TSA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Improved Ant Colony Optimization (IACO). Chapter 4 presents the application of IACO to solving the network design problem and its procedures including illustrations of examples and simulation results. Finally, Chapter 5 makes a conclusion of the research.

CHAPTER 2

Network Topology Design and Network Reliability

2.1 Network Topology Design

2.1.1 Graph Models

A network topology, defined as physical connections among the stations, is represented by a mathematical graph composed of nodes representing the stations and edges representing the communication links.

For most of the general models, there are the possibilities of node and link failures. But in this thesis, a simplified model will be used. The perfect nodes will be considered while only the links will be allowed to fail. In some situations, communication is only in one direction between a node pair, called one-direction communication. If communication occurs in both directions between two nodes, it is called bi-directional communication. This type of communication will be considered as one of the experiment conditions.

Considering bi-direction graphs composed of L links and N nodes, the notation $G(N,\ L)$ will be used. A particular node is denoted as n_i and a particular edge is denoted as l_j. An edge is also identified by the name of a pair of connected nodes. Therefore, if j is an edge between nodes s and t, the edge is represented by $l_j = (n_s,\ n_t) = l(s,\ t)$. In Figure 2.1, for example, there are four nodes $(a,\ b,\ c,\ d)$ and six edges $(1,\ 2,\ 3,\ 4,\ 5,\ 6)$ considered as a graph where $G(N=4,\ L=6)$. The nodes n_1, n_2, n_3 and are a, b, c and d, respectively. The edge 1 is denoted by $l_1 = l(n_1,\ n_2) = (a,\ b)$, the edge 2 denoted by $l_2 = l(n_2,\ n_3) = (b,\ c)$ and so on. All edges are bi-directional links. The total number of edges in the graph with n nodes is the number of combinations of n things taken two at a time, as shown in the equation below;

$$\text{Total number of edges} = \frac{n!}{(2!)(n-2)!} \qquad (2.1)$$

In the figure, therefore, the total number of edges is $\dfrac{4!}{[(2!)(4-2)!]} = 6$

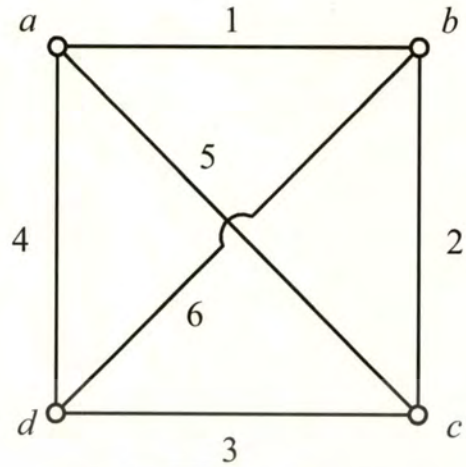


Figure 2.1 A four-node graph representing a communication network topology

In the network model, it is assumed that there are 2 states; up and down, for each link. It is also assumed that each link independently fails and no repairing or replacement of link failure is considered. Generally, the links have high reliability because of multiple (redundant) paths in all links. Thus, the network also has very high reliability. But the number of parallel paths causes high complexity in the network as well. Obviously, the efficient calculation of network reliability is a major problem in terms of analysis, design and synthesis of a communication network.

2.1.2 Mathematical Model of Communication Network Design

Notations

L	Set of possible links
l_{ij}	Options of each link
d_{ij}	Distance between node i and node j
N	Set of given nodes
n	Number of nodes
$p(l_k)$	Reliability of link option
$c(l_k)$	Unit cost of link option
x	Architecture of network design
$C(x)$	Total cost of network design
$R(x)$	Reliability of network design
R_o	Minimum network reliability constraint

In both source-destination and all-terminal network design, the reliability is the problem of network considered. For the network having a set of N nodes with specified topologies interpreted as Euclidean distance between coordinates on a plane, it represents only the cost of connection between two nodes regardless of connection type and/or level. The nodes are assumed to be fully reliable and not to fail under any circumstance. There is a set of L links which connect all N nodes. In this problem, it is assumed that every possible link is a member of L, as a fully connected network. For any (n_i, n_j) pair of N elements, hence, there is a possibility of l_{ij} element in L which l_{ij} is a connection between n_i and n_j. In addition, a link is bi-directional if l_{ij} is turned on and only one link allowed from one node to others. The search space of candidate solution related to the number of the possible links can be found by Eq. 2.2.

$$|L| = \frac{|N| \cdot |N-1|}{2} \tag{2.2}$$

In general, there are possibly more than two states in each link. For example, k connection level is given; $l_{ij} = 0, 1, 2, \ldots, k-1$. If the link is disconnected, the connection level is 0; $l_{ij} = 0$, $p(l_{ij} = 0) = 0$ is the reliability and $c(l_{ij} = 0) = 0$ is the cost per unit of $l_{ij} = 0$. If the link exists, the connection level can be 1; $l_{ij} = 1$, $p(l_{ij} = 1)$ is the reliability and $c(l_{ij} = 1)$ is the cost per unit of $l_{ij} = 1$. Therefore, various connection levels; $l_{ij} = k$, are possible for the link between node n_i and n_j where k is the connection level of the link. The mathematical formulation of this problem, when minimizing cost subject to minimum value of network reliability constraint, is;

$$\textit{Minimize} \qquad C(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{i,j} \cdot l_{i,j} \cdot d_{ij} \tag{2.3}$$

$$\textit{Subject to} \qquad R(x) \geq R_0 \tag{2.4}$$

The cost of a specific architecture, x, is given by $C(x)$ and the reliability of x is given by $R(x)$. The problem is to find a specific architecture x which minimizes the cost of connection subject to $R(x) \geq R_0$ based on the following assumptions:

1) The locations of all the network nodes are given.

2) The cost c_{ij} and the operation probability p_{ij} of each link (i, j) are fixed.

3) All links are bi-directional.

4) No redundant link is allowed in the network.

2.2 Network Reliability

The definition of reliability is the probability of the successful system operation during a given period under controlled environment. The environment includes not only temperature, atmosphere and weather but also system load and traffic quantity. Focus on communication between a pair of nodes where s is the source node and t is the target node, a successful operation is defined as the presence of one or more operating paths between s and t. This is called a two-terminal communication network and the probability of successful communication between s and t is called two-terminal reliability. If a successful operation is defined as communication ability of one node to others, it is called the all-terminal communication network. It means that node s must be able to communicate with all other $n-1$ nodes. The probability of successful communication between node s and nodes t_1, $t_2,\ldots,$ t_{n-1} is called the all-terminal reliability.

Reliability, R, is the probability of a successful operation.

In a two-terminal communication network, two-terminal reliability, R_{st}, is

$$R_{st} = P \text{ (where nodes } s \text{ and } t \text{ are connected)}$$

and in an all-terminal communication network, all-terminal reliability, R_{all}, is

$$R_{\text{all}} = P \text{ (where all } n \text{ nodes are connected)}$$

2.2.1 Two-Terminal Reliability

The evaluation of network reliability is difficult to solve and there are several approaches to achieve this. For any practical problem of significant size, a computational program must take a role. Thus all techniques mentioned in this thesis, which use “pencil-paper and calculator” analysis are introduced to an understanding of how to write algorithms and programs for network reliability computation. In this section, the following methods, namely, on event-space enumeration, cut-set and tie-set, and graph transformations will be presented.

2.2.1.1 Event-Space Enumeration for Two-Terminal Reliability

Conceptually, the simplest meaning of evaluating the two-terminal reliability of a network is to enumerate all possible combinations where each of the e graphs can be successful or failing, resulting in 2^e combinations. Each of these combinations of the successful edges and the failing edges can be treated as an event, E_i. These events are all mutually exclusive (disjoint) and the reliability expression is the probability of the union of the subset of these events that are in a path between s and t;

$$R_{st} = P(E_1 \cup E_2 \cup E_3 \ldots) \quad (2.5)$$

Since each of these events is mutually exclusive, the probability of the union becomes the sum of the individual event probabilities;

$$R_{st} = P(E_1) + P(E_2) + P(E_3) + \cdots \quad (2.6)$$

For an example of a complet four-node communication network shown in Figure 2.1, there are six edges in the graph. Considering two-terminal reliability for a node pair; a and b, where $s = a$ and $t = b$, all events associated with this graph will be $2^6 = 64$ presented in Table 2.1. The following definitions will be used for constructing the table;

E_i = the event i

j = the successful of graph

j' = the failing of graph

Table 2.1 The event-space for the graph of Figure 2.1

No failure:	$\binom{6}{0}=\frac{6!}{0!6!}=1$
$E_1 = 123456$	Successful
One failure:	$\binom{6}{1}=\frac{6!}{1!5!}=6$
$E_2 = 1'23456$	Successful
$E_3 = 12'3456$	Successful
$E_4 = 123'456$	Successful
$E_5 = 1234'56$	Successful
$E_6 = 12345'6$	Successful
$E_7 = 123456'$	Successful
Two failures:	$\binom{6}{2}=\frac{6!}{2!4!}=15$
$E_8 = 1'2'3456$	Successful
$E_9 = 1'23'456$	Successful
$E_{10} = 1'234'56$	Successful
$E_{11} = 1'2345'6$	Successful
$E_{12} = 1'23456'$	Successful
$E_{13} = 12'3'456$	Successful
$E_{14} = 12'34'56$	Successful
$E_{15} = 12'345'6$	Successful
$E_{16} = 12'3456'$	Successful
$E_{17} = 123'4'56$	Successful
$E_{18} = 123'45'6$	Successful
$E_{19} = 123'456'$	Successful
$E_{20} = 1234'5'6$	Successful
$E_{21} = 1234'56'$	Successful
$E_{22} = 12345'6'$	Successful

Table 2.1 (Continued)

Three failures:	$\binom{6}{3}=\frac{6!}{3!3!}=20$
E_{23} = 1234'5'6'	Successful
E_{24} = 123'45'6'	Successful
E_{25} = 123'4'56'	Successful
E_{26} = 123'4'5'6	Successful
E_{27} = 12'345'6'	Successful
E_{28} = 12'34'56'	Successful
E_{29} = 12'34'5'6	Successful
E_{30} = 12'3'456'	Successful
E_{31} = 12'3'45'6	Successful
E_{32} = 12'3'4'56	Successful
E_{33} = 1'2345'6'	Successful
E_{34} = 1'234'56'	Successful
E_{35} = 1'234'5'6	Failing
E_{36} = 1'2'3456'	Failing
E_{37} = 1'2'345'6	Successful
E_{38} = 1'2'34'56	Successful
E_{39} = 1'23'456'	Successful
E_{40} = 1'23'45'6	Successful
E_{41} = 1'23'4'56	Successful
E_{42} = 1'2'3'456	Successful

Table 2.1 (Continued)

Four failures:	$\binom{6}{4}=\frac{6!}{4!2!}=15$
$E_{43}=123'4'5'6'$	Successful
$E_{44}=12'34'5'6'$	Successful
$E_{45}=12'3'45'6'$	Successful
$E_{46}=12'3'4'56'$	Successful
$E_{47}=12'3'4'5'6$	Successful
$E_{48}=1'234'5'6'$	Failing
$E_{49}=1'23'45'6'$	Failing
$E_{50}=1'23'4'56'$	Successful
$E_{51}=1'23'4'5'6$	Failing
$E_{52}=1'2'345'6'$	Failing
$E_{53}=1'2'34'56'$	Failing
$E_{54}=1'2'34'5'6$	Failing
$E_{55}=1'2'3'456'$	Failing
$E_{56}=1'2'3'45'6$	Successful
$E_{57}=1'2'3'4'56$	Failing

Five failures:	$\binom{6}{5}=\frac{6!}{5!1!}=6$
$E_{58}=12'3'4'5'6'$	Successful
$E_{59}=1'23'4'5'6'$	Failing
$E_{60}=1'2'34'5'6'$	Failing
$E_{61}=1'2'3'45'6'$	Failing
$E_{62}=1'2'3'4'56'$	Failing
$E_{63}=1'2'3'4'5'6$	Failing

Six failures:	$\binom{6}{6}=\frac{6!}{6!0!}=1$
$E_{64}=1'2'3'4'5'6'$	Failing

The term "*successful*" means that there is at least one path connecting a and b in the given combination of successful and failing edges. Term "*failing*", on the other hand, means that there is no path which connects a and b in the given combination of successful and failing edges. The result, successful or failing, is determined by an inspection of the graph.

Substitution of the "successful" events from Table 2.1 in Eq. (2.6) yields the two-terminal reliability from a to b;

$$\begin{aligned} R_{ab} &= [P(E_1)] + [P(E_2) + \cdots + P(E_7)] + [P(E_8) + P(E_9) + \cdots + P(E_{22})] \\ &\quad + [P(E_{23}) + P(E_{24}) + \cdots + P(E_{34}) + P(E_{37}) + \cdots + P(E_{42})] \\ &\quad + [P(E_{43}) + P(E_{44}) + \cdots + P(E_{47}) + P(E_{50}) + P(E_{56})] + [P(E_{58})] \end{aligned} \tag{2.7}$$

There is one term in the first bracket in Eq. (2.7) where all of the edges must be Successful. If all edges are identical and independent and have a probability of success, p, then the probability of event E_1 is p^6. Similarly, for the second bracket, there are six events of success probability and probability of failure is q, then the probability of event E_2 is qp^5 where the probability of failure $q = 1 - p$. Substitution of p and q in Eq. (2.7) yields a polynomial;

$$R_{ab} = p^6 + 6qp^5 + 15q^2p^4 + 18q^3p^3 + 7q^4p^2 + q^5p \tag{2.8}$$

Numerical evaluation of the polynomial for $p = 0.9$ and $q = 0.1$ yields;

$$\begin{aligned} R_{ab} &= 0.9^6 + 6(0.1)(0.9)^5 + 15(0.1)^2(0.9)^4 + 18(0.1)^3(0.9)^3 \\ &\quad + 7(0.1)^4(0.9)^2 + (0.1)^5(0.9) = 0.997848 \end{aligned} \tag{2.9}$$

Usually, event-space reliability calculations require much effort and time even though the procedure is clear. The number of events builds up exponentially as 2^e. For $e = 10$, the number of all terms is 1,024 and there will be over a million terms if the value of e is twice.

2.1.1.2 Cut-Set and Tie-Set Methods for Two-Terminal Reliability

To reduce the amount of work in a network reliability analysis lower than the 2^e complexity required for the event-space method, the minimal cut sets and minimal tie sets of a graph have been introduced. The tie sets are the groups of edges that form a path between s and t. The term "minimal" implies that no node or edge is traversed more than once, but another way of defining is that minimal tie sets have no subset of edges which is a tie set. If there are i tie sets between s and t, then the reliability expression is given by the expansion of;

$$R_{st} = P(T_1 \cup T_2 \cup \ldots \cup T_i) \tag{2.10}$$

Similarly, on the minimal cut sets of a graph, the cut sets are groups of edges that break all paths between s and t when those edges are removed from the graph. If a cut set is minimal, no subset is in a cut set. The reliability expression in terms of j, cut sets is given by the expansion of;

$$R_{st} = 1 - P(C_1 \cup C_2 \cup \ldots \cup C_3) \tag{2.11}$$

Applying the above theory to the example in Figure 2.1, the minimal cut sets and tie sets are found by an inspection of $s = a$ and $t = b$ given in Table 2.2. Since there are fewer cut sets, it is easier to use Eq. (2.11) rather than Eq. (2.10). However, there is no common rule when $j < i$ or vice versa.

Table 2.2 Minimal tie sets and cut sets for the example of Figure 2.1 ($s = a, t = b$)

Tie Sets	Cut Sets
$T_1 = 1$	$C_1 = 1'4'5'$
$T_2 = 52$	$C_2 = 1'6'2'$
$T_3 = 46$	$C_3 = 1'5'6'3'$
$T_4 = 234$	$C_4 = 1'2'3'4'$
$T_5 = 536$	-

$$\begin{aligned}
R_{ab} &= 1 - P(C_1 \cup C_2 \cup C_3 \cup C_4) \\
R_{ab} &= 1 - P(1'4'5' + 1'6'2' + 1'5'3'6' + 1'2'3'4') \\
R_{ab} &= 1 - [P(1'4'5') + P(1'6'2') + P(1'5'3'6') + P(1'2'3'4')] \\
&\quad + [P(1'2'4'5'6') + P(1'3'4'5'6') + P(1'2'3'4'5') \\
&\quad + P(1'2'3'5'6') + P(1'2'3'4'6') + P(1'2'3'4'5'6')] \\
&\quad - [P(1'2'3'4'5'6') + P(1'2'3'4'5'6') + P(1'2'3'4'5'6') \\
&\quad + P(1'2'3'4'5'6')] + [P(1'2'3'4'5'6')]
\end{aligned} \tag{2.12}$$

The probability expansion of a union of events that occurs in Eq. (2.12) is often called the inclusion-exclusion formula. Note that in the expansion of Eq. (2.10) or Eq. (2.11), it is based on the theorem of $x \cap x = x$ and $x \cup x = x$. For example, in the second bracket of Eq. (2.12), its second term is $P(C_1C_3) = P([1'4'5'][1'5'6'3']) = P(1'3'4'5'6')$ since $1' \cap 1' = 1'$ and $5' \cap 5' = 5'$.

If all of the edges have equal probability of failure $= q$ and are independent, Eq. (2.12) becomes;

$$R_{ab} = 1-[2q^3+2q^4]+[5q^5+q^6]-[4q^6]+[q^6]$$
$$R_{ab} = 1-2q^3-2q^4+5q^5-2q^6 \quad (2.13)$$

Substitution of $p = 0.9$ and $q = 0.1$ in Eq. (2.13) yields;

$$R_{ab} = 1-2\times(0.1)^3-2\times(0.1)^4+5\times(0.1)^5-2\times(0.1)^6 = 0.997848 \quad (2.14)$$

The result of Eq. (2.14) is identical to Eq. (2.9). The expansion of Eq. (2.11) has a complexity of 2^j, which is more complex than in Eq. (2.10) if there are more cut sets than tie sets. Due to this point, the network should be analyzed and investigated how many tie sets and cut sets exist between nodes s and t.

2.1.1.3 Cut-Set and Tie-Set Approximations for Two-Terminal Reliability

The inclusion-exclusion expansions of Eq. (2.10) and Eq. (2.11), sometimes, yield a sequence of probabilities which decrease size so that many terms of higher order in the sequence can be neglected, resulting in a simple approximate formula. These terms are products of probabilities, so if these probabilities are small, the higher-order product terms can be neglected. The cut-set approximations are the most frequently used in practice. If only the first bracket in Eq. (2.12) is retained in addition to the unity term, it is called the retention of only the first two terms of the disjoint approximation.

For the example in Figure 2.1, Eq. (2.15) has been obtained for the disjoint approximation; assume that $q = 0.1$

$$R_{ab} \geq 1-[2q^3+2q^4] = 1-0.002-0.0002 = 0.9978 \quad (2.15)$$

which is quite close to the exact value given in Eq. (2.14). If the next bracket is included in Eq. (2.12), the approximation will be closer at the expense of computing $[j+\binom{j}{2}]=[j(j-1)/2]$ terms.

$$\begin{aligned} R_{ab} &\leq 1-[2q^3+2q^4]+[5q^5+q^6] \\ &= 0.9978+5\times 0.1^5+0.1^6=0.997851 \end{aligned} \tag{2.16}$$

Eq. (2.16) is not only an approximation but also an upper bound. In fact, as more terms are included in the inclusion-exclusion formula, a set of alternative bounds will be obtained. Note that Eq. (2.15) is a sharp lower bound and Eq. (2.16) is even sharper, but both equations have an effective bracket to the results. Clearly, the sharpness of these bounds increases as $q_i = 1-p_i$ decreases for the i edges of graph.

$$0.997800 \leq R_{ab} \leq 0.997851 \tag{2.17}$$

R_{ab} can be approximated by the midpoint of two bounds.

$$R_{ab}=\frac{0.997800+0.997851}{2}=0.9978255 \tag{2.18}$$

The accuracy of the preceding approximation can be evaluated by examining the deviation in the computed probability of failure $F_{ab}=1-R_{ab}$. In the region of high reliability, all the values of R_{ab} are very close to unity and differences are misleadingly small. Thus, as an error criterion, the following equation will be used;

$$\%error=\frac{|F_{ab}(estimate)-F_{ab}(exact)|}{F_{ab}(exact)}\times 100 \tag{2.19}$$

Evaluation of Eq. (2.19) for the results in Eq. (2.14) and Eq. (2.18) yields

$$\%error=\frac{|0.0021745-0.002152|}{0.002152}\times 100=1.05 \tag{2.20}$$

2.1.2 All-Terminal Reliability

The all-terminal reliability problem is more difficult than the two-terminal reliability problem. It's necessary to modify the two-terminal problem for all-terminal pairs. The methods of Section 2.3 will be discussed in this section for the case of all-terminal reliability.

2.1.2.1 Event-Space Enumeration for All-Terminal Reliability

All of the Successful events for two-terminal reliability will be examined and classified as Failing in case of no connection for all the terminal pairs. By applying these restrictions to Table 2.1, Table 2.3 will be obtained. From the table, an all-terminal reliability expression has been formulated similar to the two-terminal case;

$$R_{all} = \sum_{\substack{i=1 \\ i \neq 25,31,35,36}}^{42} P(E_i) \tag{2.21}$$

Note that events 25, 31, 35, and 36 represent only the failure events with three edge failures. These four cases involve isolation of each of the four vertices. All other failure events involve four or more failures. Substitution of the terms from Table 2.3 in Eq. (2.21) yields

Table 2.3 Modification of Table 2.1 for the all-terminal reliability problem

Connection *ab*	Connection *ac*	Connection *ad*	Term	Event
√	√	√	p^6	E_1
√	√	√	qp^5	$E_2, E_3, \ldots, E_7$
√	√	√	q^2p^4	$E_2, E_3, \ldots, E_7$ $E_{23}, E_{24}, E_{26}, E_{27}$ E_{28}, E_{29}, E_{30} $E_{32}, E_{33}, E_{34}, E_{37}$ $E_{38}, E_{39}, E_{40}, E_{41}$
√	√	√	q^3p^3	E_{42}
—	—	—	$q^3p^3, q^4p^2, q^3p^1, q^6$	Other 26 fail for at least 1 terminal pair

$$R_{all} = p^6 + 6qp^5 + 15q^2p^4 + 16q^3p^3$$
$$= 0.9^6 + 6\times0.1\times0.9^5 + 15\times0.1^2\times0.9^4 + 16\times0.1^3\times0.9^3$$
$$+ 0.354294 + 0.098415 + 0.011664$$
$$R_{all} = 0.995814$$

2.1.2.2 Cut-Set and Tie-Set Methods for All-Terminal Reliability

Starting with finding the tie sets and cut sets for the terminal pairs of ab, ac, and ad, there is a connection between all nodes if node a is connected to all other nodes. In terms of tie sets, the equation will be;

$$P_{all}=P([\text{path } ab].[\text{path } ad].[\text{path } ac]) \quad (2.22)$$

$$P_{all}= P([T_1 \cup T_2 \cup \cdots \cup T_5].[T_6 \cup T_7 \cup \cdots \cup T_{10}].[T_{11} \cup T_{12} \cup \cdots \cup T_{15}]) \quad (2.23)$$

Table 2.4 Cut sets and tie sets for all-terminal reliability computation

Pair *ab*	Pair *ad*	Pair *ac*
Tie Sets		
$T_1 = 1$	$T_6 = 5$	$T_{11} = 4$
$T_2 = 25$	$T_7 = 12$	$T_{12} = 53$
$T_3 = 46$	$T_8 = 43$	$T_{13} = 16$
$T_4 = 234$	$T_9 = 163$	$T_{14} = 123$
$T_5 = 356$	$T_{10} = 462$	$T_{15} = 526$
Cut Sets		
$C_1 = 1'5'4'$	$C_5 = 5'4'1'$	$C_9 = 4'6'3'$
$C_2 = 1'2'6'$	$C_6 = 5'3'2'$	$C_{10} = 4'5'1'$
$C_3 = 1'3'5'6'$	$C_7 = 5'6'4'2'$	$C_{11} = 4'1'2'3'$
$C_4 = 1'2'3'4'$	$C_8 = 5'6'1'3'$	$C_{12} = 4'2'5'6'$

The expansion of Eq. (2.23) involves 125 intersections followed by complex calculations involving the expansion of union of the resulting events (inclusion-exclusion). Clearly, manual computations are starting to become intractable. A similar set of equations can be written in terms of cut sets. In this case, disconnecting of the path ab, ad, or ac is sufficient to generate all-terminal cut sets.

P_{all}=1-P ([no path ab].[no path ad].[no path ac]) (2.24)

$$P_{all} = 1 - P([C_1 \cup C_2 \cup C_3 \cup C_4] + [C_5 \cup C_6 \cup C_7 \cup C_8] + [C_9 \cup C_{10} \cup C_{11} \cup C_{12}]) \tag{2.25}$$

The expansion of Eq. (2.25) involves the expansion of the union for 12 events ($2^{12} = 4{,}096$ terms) and the disjoint or reduced approximation or computer solution is the only practical approaches.

2.1.2.3 Cut-Set and Tie-Set Approximations for All-Terminal Reliability

The difficulty of expanding Eq. (2.23) and Eq. (2.25) makes approximations almost imperative in any pencil-paper-and-calculator analysis. By simplifying Eq. (2.25), $C_5, C_{10}, C_{11}, C_{12}$ and C_8 can be dropped, thereby terms of cut set in Eq. (2.25) are reduced to seven cut sets. Since all edges are assumed to have equal reliabilities, $p = 1 - q$, and the disjoint approximation for Eq. (2.25) yields;

$$P_{all} \geq 1 - P(C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_6 \cup C_7 \cup C_9) \tag{2.26}$$

and substituting $q = 0.1$ yields;

$$P_{all} \geq 1 - 4q^3 - 3q^4 = 1 - 4(0.1)^3 - 3(0.1)^4 = 0.9957 \tag{2.27}$$

To obtain an upper bound, the 21 terms have been added in the second bracket in the expansion of Eq. (2.25) to yield;

$$P_{all} \leq 0.9957 + 17q^5 + 4q^6 = 0.99600 \tag{2.28}$$

The average of Eq. (2.27) and Eq. (2.28) results in $P_{all} \approx 0.995759$, which is $(0.000174 \times 100 / 0.0004) = 4.21\%$ in error. In this case, the approximation yields excellent results.

CHAPTER 3

Improved Ant Colony Optimization Algorithm

This chapter introduces Improved Ant Colony Optimization (IACO) including other optimization techniques applied to network design problem in this thesis. Firstly, the principles of several techniques such as Genetic Algorithm (GA), Tabu Search Algorithm (TSA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are described. Subsequently, the motivation of the proposed algorithm, Improved Ant Colony Optimization (IACO), is described. Furthermore, the principle of IACO and the additional salient mechanisms for improvement of search process, i.e. neighborhood search and re-initialization process are explained. Finally, the feasibility study of the IACO is demonstrated to solving the benchmark optimization problems. The optimized results by IACO are compared with the conventional approaches that are mentioned above in terms of solution quality and computational efficiency.

3. 1 Genetic Algorithm (GA)

In 1975, John Holland has proposed the concept of Genetic Algorithm [14]. Based on the mechanisms of natural selection and genetics; weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction.

The GA has been started with random generation of initial population and than the selection, crossover and mutation are produced until the best population is found. They can search several possible solutions simultaneously.

3.1.1 Principle of Genetic Algorithm

In GA terminology, a solution vector is called a chromosome. Chromosomes are made of discrete units called genes. Each gene controls one or more features of the chromosome. In the original GA, genes are assumed to be binary digits. In later implementations, more varied gene types have been introduced. Normally, a chromosome corresponds to a unique solution in the solution space. This requires a mapping mechanism between the solution space and the chromosomes. This mapping is called an encoding.

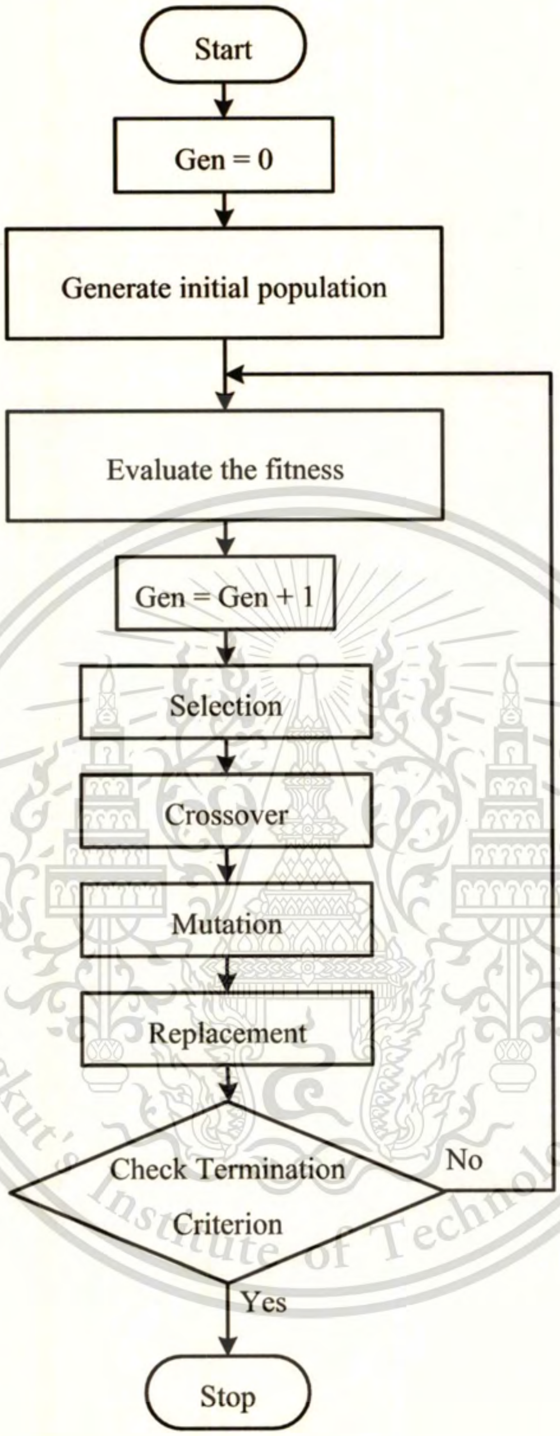


Figure 3.1 A flow chart of a Genetic Algorithm

Genetic Algorithm presented in Figure 3.1, which is described as follows;

Step 1 *Initiation;* Start with a randomly generated population "n" strings.

Step 2 *Evaluation;* Calculate the fitness function of each string in the population.

Step 3 *Re-production;* Select a pair of parent strings from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement" meaning that the same string can be selected more than once to become a parent.

Step 4 *Crossover;* With the crossover probability, cross over the pair at a randomly chosen point to form two new strings. If no crossover takes place, form two new strings that are exact copies of their respective parents.

Step 5 *Mutation;* Mutate the two new strings at each locus with the mutation probability, and place the resulting strings in the new population.

Step 6 *Replacement;* Replace the current population with the new population.

Step 7 *Repeat algorithm;* Go to step 2.

3.1.2 Components of Genetic Algorithm

3.1.2.1 Chromosome Encoding

Coding involves representing each element of a population into a string of values known as a chromosome. There are a couple of ways to code a population.

Some methods are listed below:

a) Binary string

b) Decimal

c) Floating point

Of all the methods listed above, binary coding is the most popular as anything can be adequately represented in binary bits. On top of that, crossover and mutation operations are more easily carried out. For binary representation, the length of ht string determines the resolution of the coding. Longer strings give better resolution but down the speed of calculation involved.

Binary string example presented below:

Chromosome A: 1 0 0 0 0 1 1 1 0 0

Chromosome B: 1 1 0 0 0 1 0 0 0 1

3.1.2.2 Initial Population

Initial population is randomly generated before starting genetic procedure (presented as Figure

3.2) so that there is less chance concentration of values. The larger the size of the initial population, the longer the GA would run. However, for large population, the optimized solution is expected within fewer generations.

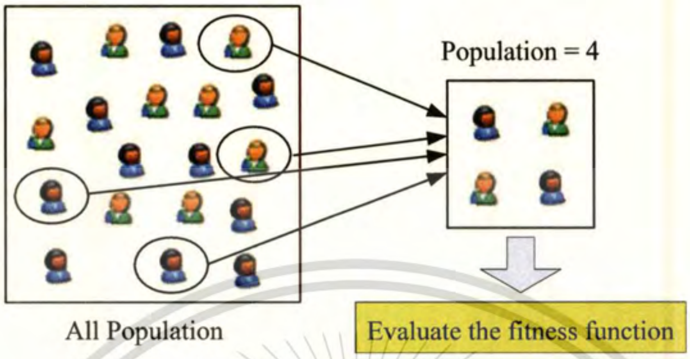


Figure 3.2 Randomly generated 4 initial population

3.1.2.3 Fitness Function

Fitness function is one of the key features of GA. It depends on the objective function that is being optimized. Thus, better individuals (higher fitness values) must be those having the lower values of the objective function.

3.1.2.4 Genetic Operator

3.1.2.4.1 Crossover

During crossover, random groups are swapped. The algorithms code crossover as a swap of several bits (used with genetic programming) and a distinction is made between two parents (bit strings, but called chromosomes in GA terminology) who are the identical, two different parents and single parent.

The process has the following procedure and an example presented in Figure 3.3:

1. Select two chromosomes.
2. Cut the chromosome (or branch) at a particular location.
3. Swap the bits/branches of the two parents.

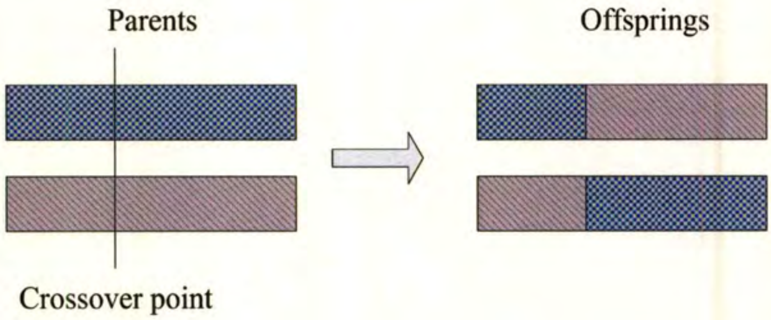


Figure 3.3 Crossover

3.1.2.4.3 Mutation

Crossover can generate a very large amount of different strings. However, depending on the initial population chosen, there may not be enough variety in the strings to ensure the GA covers the entire problem space. On the other hand, the GA may find itself converging on strings that are not quite close to the optimum it seeks due to a bad initial population. Some of these problems are overcome by introducing a mutation operator into the GA. The GA has a mutation probability, which dictates the frequency at which mutation occurs. Mutation can be performed during either selection or crossover. For each string element in each string in the mating pool, the GA checks to see if it should perform a mutation. If it should, it randomly changes the element value to a new one. In our binary strings, 1s are changed to 0s and 0s to 1s as presented in Figure 3.4

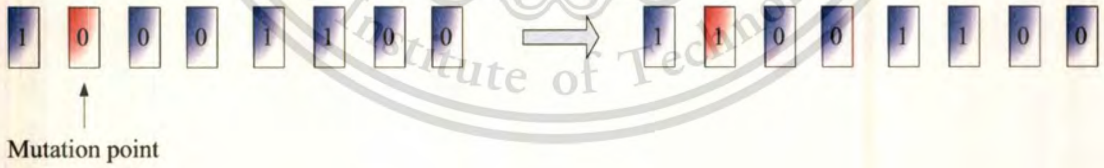


Figure 3.4 Mutation

3.2 Tabu Search Algorithm (TSA)

Fred Glover first presented Tabu Search in its present form in 1986 [15, 16]. Many computational experiments have presented that Tabu Search has now become an established optimization technique, which can compete with almost all known techniques and can beat many classical procedures. Up to now, there is no formal explanation of this good behavior.

3.2.1 Principle of Tabu Search Algorithm

Tabu Search is an iterative search method. It uses a local search algorithm at each iteration to search for the best solution in some subset of the neighborhood, which came from the best solution obtained at the last iteration. At each iteration, the local search algorithm looks for the best improving solution. If no solutions are improving the objective function value, then it looks for the least deterioration solution. Tabu Search keeps a list, which is called Tabu List, of the moves it used to obtain the best solutions during each iteration and to restrict the local search algorithm in reusing those moves. By using the Tabu List, Tabu Search enables the acceptance of non-improving solutions to escape from local optimum. However, there is an important exception to the Tabu List. If the Tabu moves generate a better solution than all the best solutions obtained so far, Tabu moving will be accepted and free it out of the Tabu List. This override of the Tabu List is called "Aspiration Conditions"

Tabu Search algorithm presented in Figure 3.5, which described as follows:

Step 1: Randomly generate an initial solution, s_0, in search space, S. and set number of iteration, i=0.

Step 2: Set i=i+1, the best solution, $s_{best} = s_0$ and the best cost, $c_{best} = f(s_{best})$ where $f(\cdot)$ is fitness function.

Step3: Check conditions in Tabu List (TL), if conditions meet restrictions then reset Tabu List.

Step 4: Generate a set of neighborhood of solution, s_n*, in S. Evaluate Fitness function of s_n*, $f(s_n*)$ and choose the best solution which is NOT in the TL, then accept it as a current solution, s_{n_best} and set $c_{n\_best} = f(s_{n\_best})$.

Step 5: Check if $f(s_{n\_best}) < f(s_0)$ then $s_0 = s_{n\_best}$ else $s_0 = s_{best}$.

Step 6: Update the Tabu List and Aspiration conditions.

Step 7: If a termination condition is met then stop. Else, go to Step 3.

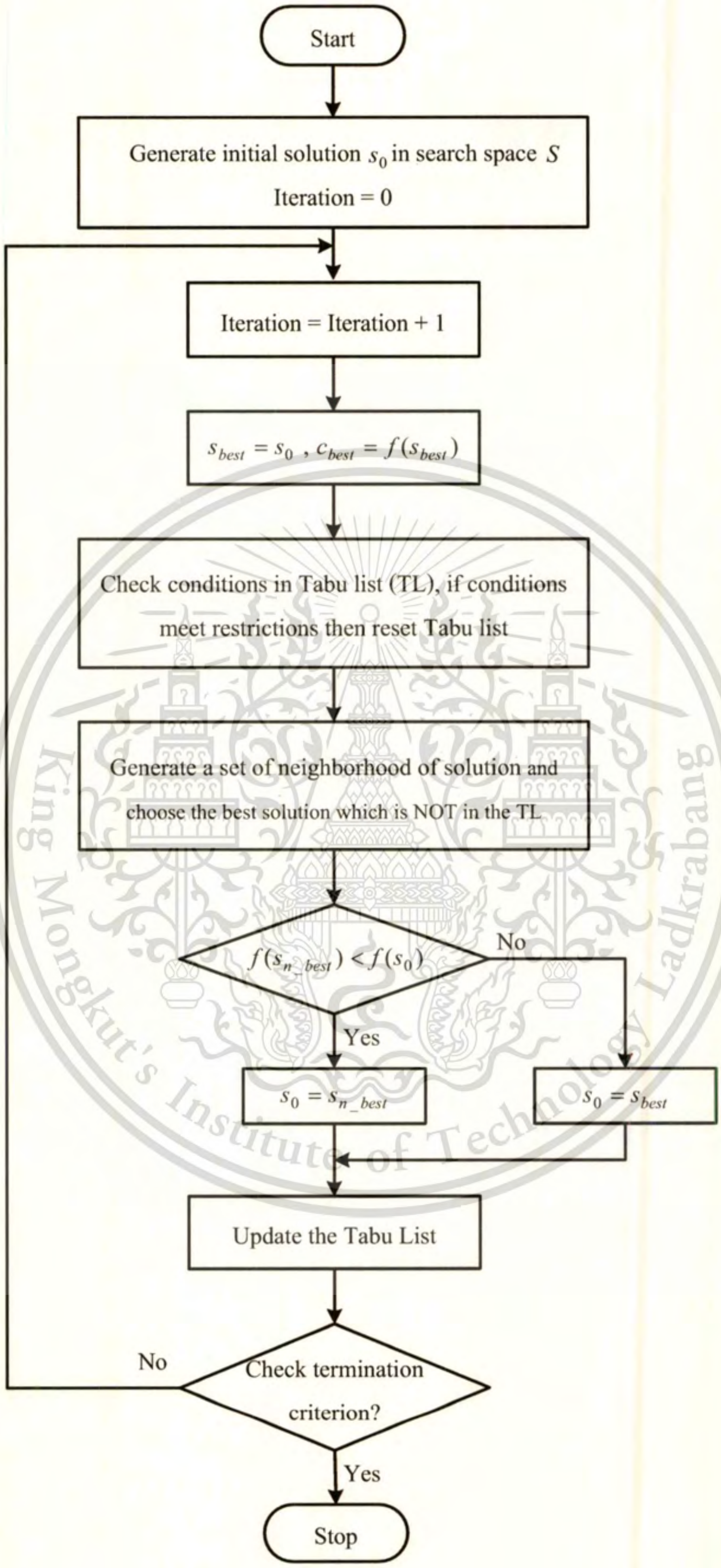


Figure 3.5 A flow chart of a Tabu Search algorithm

3.2.2 Components of Tabu Search Algorithm

3.2.2.1 Tabu List (TL)

The basic role of the Tabu List is to prevent cycling. If the length of the list is too small, this role might not be achieved; conversely a too long size creates too many restrictions and it has been observed that the mean value of the visited solutions grows with the increase of the Tabu List size. Usually, an order of magnitude of this size may be easily determined. However, given an optimization problem it is often difficult or even impossible to find a value that prevents cycling and does not excessively restrict the search for all instances of the problem of a given size. An effective way for circumventing this difficulty is to use a Tabu List with variable size. Each element of the list belongs to it for a number of iterations that is bounded by given maximal and minimal values.

Proposed Tabu List in this thesis composed of 2 elements which are solution recorder and Tabu status recorder. Solution recorder contains the best solution of search which has been presented in Figure 3.6. Tabu status recorder contains all the Tabu moves that cannot be applied to the present solution. The moves stored in the Tabu List are those carried out most frequency, recency and Tabu status according to some criteria called Tabu restrictions which has been presented in Figure 3.7. The use of Tabu List decreases the possibility of cycling because it prevents returning in a certain number of iterations to a solution visited recently

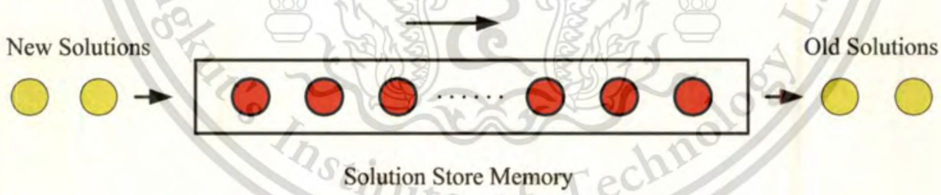


Figure 3.6 The structure of solutions storage

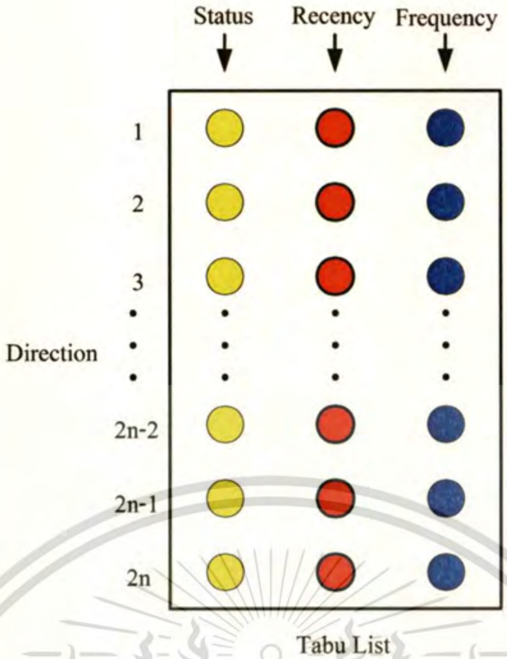


Figure 3.7 Structure of Tabu List

3.2.2.2 Aspiration Criterion

Tabu restrictions might sometimes prevent the search to find the solutions which have not been visited yet or even cause all available moves to be classified as the Tabu. Therefore, it should be possible to forget the Tabu constraints when a freedom is required for the search. A criterion called aspiration criteria is employed to determine which moves should be freed in such cases.

Aspiration Criterion encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Again, such an approach can be based on generating subassemblies of solution components that can rely on modified evaluations as embodied. For example, in the use of Aspiration Criterion, which is membership in the set, is often determined by setting a threshold, which is connected to the objective function value of the best solution found during the search.

3.3 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) is parallel stochastic search algorithm first proposed by Kennedy and Eberhart in 1995 [17]. It is affected from a wide diversity study of social animals in nature discovered their interesting behaviors such as fish schooling and a flock of birds. In particular, the researches focus on collective behavior that result from the local interactions of the individuals with each other and with their environment. Agents in a swarm act relatively independently from all others in order to achieve the goal of swarm they belong without following commands or rules [18].

3.3.1 Principle of Particle Swarm Optimization

PSO has been developed through simulation of simplified social models. It is based on a simple concept. Therefore, the computation time is short and it requires few memories. The system is initialized with a population of random solutions. In each potential solution is assigned a randomized velocity, and the potential solutions, called particles. Principle of Particle Swarm Optimization has been presented in Figure 3.8

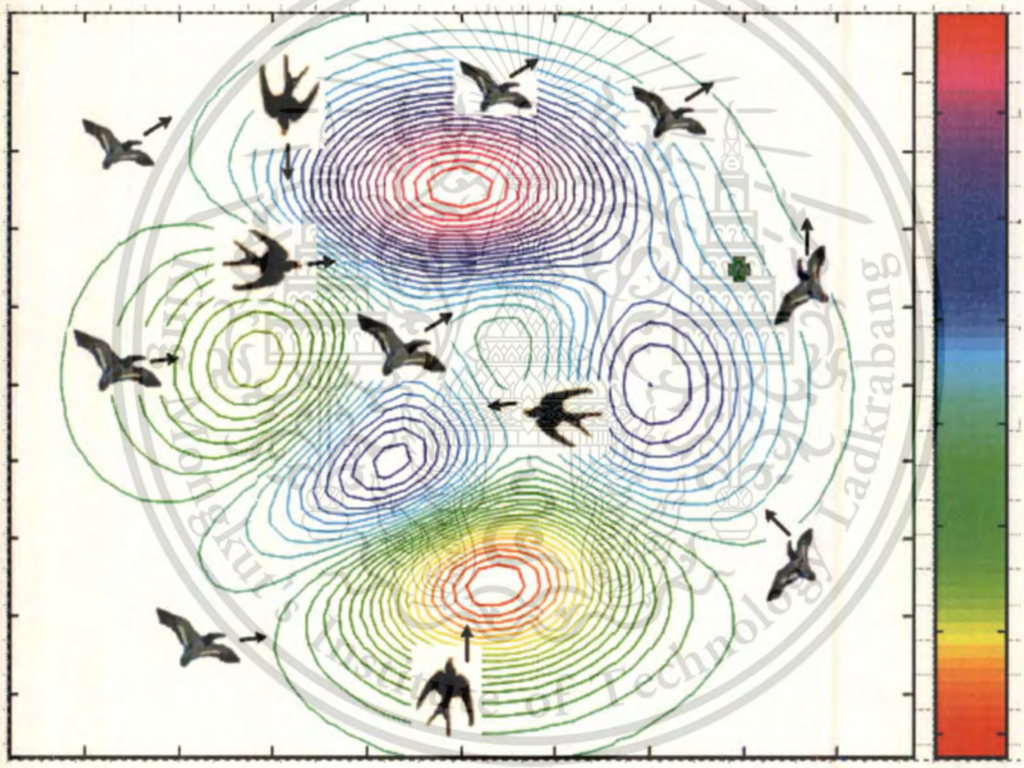


Figure 3.8 Principle of Particle Swarm Optimization

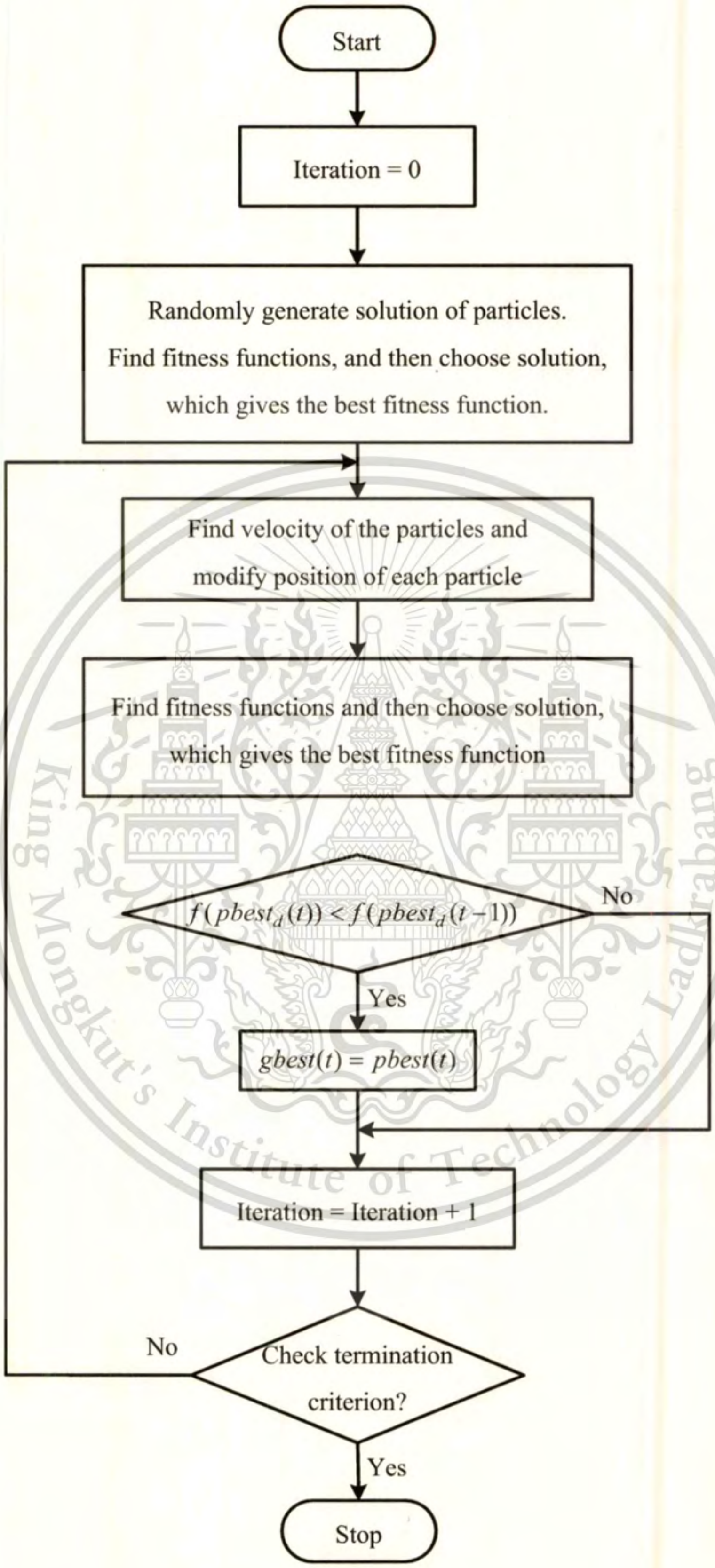


Figure 3.9 A flow chart of a Particle Swarm Optimization

Particle Swarm Optimization presented in Figure 3.9, which described as follows:

Definition of parameters

m Number of particle

x_i Solution of particle i

v_i Velocity of particle i

$pbest_i$ Solution of a particle i, which find the best solution in each iteration

$gbest_i$ Solution of a particle i, which find the best solution in all iterations

Step 1: Set number of iteration, $i = 0$. Randomly generate solution $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ of m particles. Find fitness functions, $F = f(x_i) = (f(x_{i1}), f(x_{i2}), ..., f(x_{im}))$ and then choose solution, which gives the best fitness function. Set $gbest = x_{im}$.

Step 2: Find velocity of the particles, $v_i = (v_{i1}, v_{i2}, ..., v_{im})$ using Eq. (3.1) and modify position of each particle using Eq. (3.2).

Step 3: Find fitness functions, $F = f(x_i) = (f(x_{i1}), f(x_{i2}), ..., f(x_{im}))$ and then choose solution, which gives the best fitness function. Set $pbest_{im} = x_{im}$.

Step 4: Check if $f(pbest_{im}) < f(pbest_{im-1})$ then $pbest_m = pbest_{im}$. Check if $f(gbest) < f(pbest_m)$ then $gbest = pbest_m$.

Step 5: Set Iteration = Iteration + 1

Step 6: If a termination condition is met then stop. Else, go to Step 2.

3.3.2 Components of Particle Swarm Optimization

The modified velocity of each agent can be calculated using the current velocity and the distance from $pbest_{id}$ and $gbest_d$ as shown below:

$$v_{id}^{t+1} = \omega \cdot v_{id}^t + [c_1 \cdot rand(\cdot) \cdot (pbest_{id} - x_{id}^t)] + [c_2 \cdot rand(\cdot) \cdot (gbest_d - x_{id}^t)] \quad (3.1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (3.2)$$

where v_{id}^t current velocity of agent at iteration, $V_d^{\min} \le v_{id}^t \le V_d^{\max}$

v_{id}^{t+1} modified velocity of agent

$rand(\cdot)$ random number between 0 and 1

x_{id}^t current position of agent at iteration

$pbest_{id}$ *pbest* of agent

$gbest_d$ *gbest* of the group

ω weight function for velocity of agent

c_1, c_2 weight coefficients for each term

3.4 Ant Colony Optimization (ACO)

ACO is an algorithm that was inspired by the behavior of real ants. Entomologists have studied how blind animals such as ants are capable of finding the shortest path from food sources to the nest without using visual cues. They are also capable of adapting to changes in the environment, for example, finding a new shortest path once the old one is no longer feasible due to a new obstacle. The studies of entomologists reveal that such capabilities are essentially due to communicating information among individuals regarding path used to decide the direction. Ants deposit a certain amount of pheromone while walking and each ant probabilistically prefers to follow a direction rich in pheromone rather than that with less amount.

The process can be clearly illustrated by Figure 3.10 (a), in which ants are on a straight line that connects a food source to their nest. An ant will deposit pheromone while walking and it probabilistically prefers to follow a direction rich in pheromone.

This behavior can explain how ants can find the shortest path that reconnects a line that is broken by an obstacle in Figure 3.10 (b). Those ants are just in front of the obstacle and they cannot continue to go. Therefore, they have to choose between turning right or left. Half the ants choose to turn right and the other half choose to turn left. A similar situation arises on the other side of the obstacle as seen in Figure 3.10 (c). Ants choose the shorter path will more rapidly reconstitute the interrupted pheromone trail compared with those choosing the longer path. Thus, the shorter path will receive a greater amount of pheromone per time unit and, in turn, a lager number of ants will choose the shorter path. Due to this positive feedback, all the ants will rapidly choose the shorter path as revealed in Figure 3.10 (d). All ants move approximately the same speed and deposit a pheromone trail at approximately the same rate. The time consumed on the longer side of an obstacle is greater than the shorter path. It thus makes the pheromone trail accumulate more quickly on the shorter path. Moreover, ants prefer the higher number of pheromone trails that enable accumulation to can be built up faster on the shorter path.

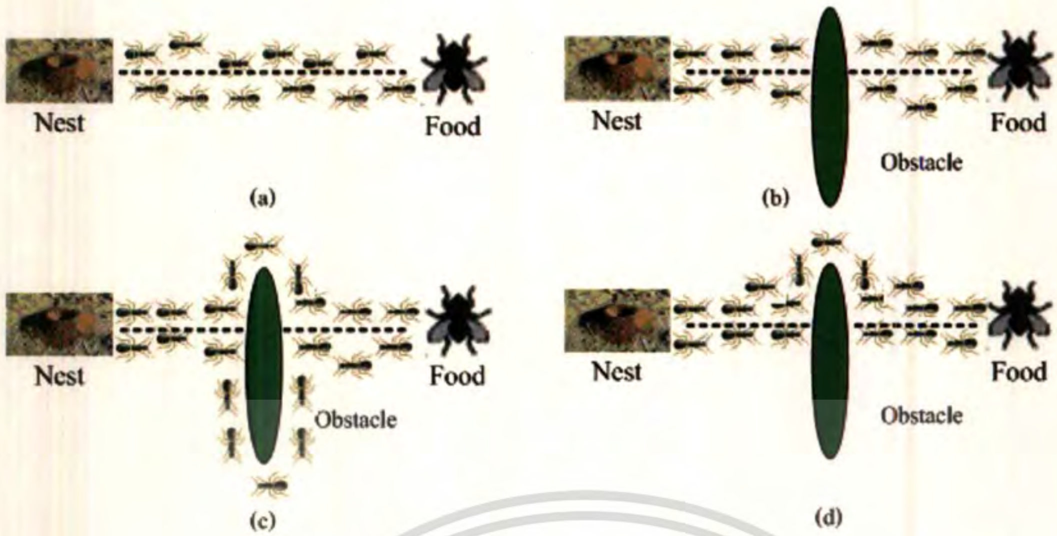


Figure 3.10 Behavior of ants;

(a) Real ants follow a path between nest and food source

b) An obstacle appears on the path: ants choose whether to turn left or right with equal probability

(c) Pheromone is deposited more quickly on the shorter path

(d) All ants have chosen the shorter path

3.4.1 Principle of Ant Colony Optimization

Inspired by the collective behavior of a real ant colony, Marco Dorigo first introduced the Ant System (AS) in his Ph.D. thesis (1992) and Dorigo et al. further continued the study [19]. The characteristics of an artificial ant colony include positive feedback, distributed computation and the use of a constructive greedy heuristic. Positive feedback accounts for rapid discovery of good solutions, distributed computation avoids premature convergence and the greedy heuristic helps to find acceptable solutions in the early stages of the search process. In order to demonstrate the AS approach, the authors apply this approach to the classical TSP, asymmetric TSP, quadratic assignment problem (QAP) and job-shop scheduling. The AS shows very good results in each applied area. More recently, Dorigo and Gambardella have worked on extended versions of the AS paradigm. ACO is the one of the extensions and has been applied to the symmetric and asymmetric TSP with excellent results [20]. The Ant System has also been applied with success to other combinatorial optimization problems such as the vehicle routing problem [21, 22].

In general, the procedure of an ACO algorithm can be described as follows: m ants are initially positioned at the nest. Each ant will choose a possible route as a solution. Ants are guided, in building their tours, by both heuristic information and pheromone information. The amount of pheromone is modified by applying the global updating rule. Naturally, a link with a high amount of pheromone is a desirable choice. The pheromone updating rules are designed so that they tend to give more pheromone to edges, which should be visited by ants.

In fact, each ant builds a feasible solution (called a tour) by a stochastic greedy search called the state transition rule, which is repeatedly applied.

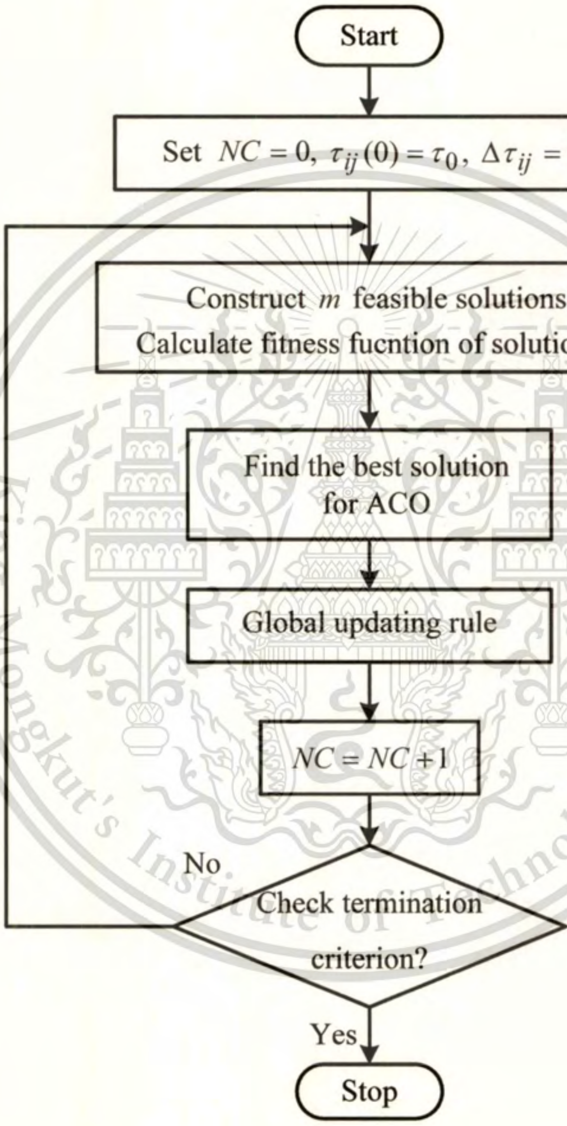


Figure 3.11 A flow chart of an Ant Colony Optimization algorithm

A flow chart of an Ant Colony Optimization algorithm is presented in Figure 3.11 which described as follows:

Step 1 Initialization

Set NC = 0 /* NC: cycle counter */

For every combination (i, j)

Set an initial value $\tau_{ij}(0) = \tau_0$ and $\Delta\tau_{ij} = 0$

End

Step 2 Construct feasible solutions

For k=1 to m /* m: number of ants */

For i=1 to n /* n: number of links */

Choose a level of connection with transition probability given by Eq. (3.4)

End

Calculate fitness function f_k /* f_k: fitness value of each ant */

End

Update the best solution

Step 3 Global updating rule

For every combination (i, j)

For k=1 to m

Find $\Delta\tau_{ij}^k$ according to Eq. (3.8)

End

Update $\Delta\tau_{ij}$ according to Eq. (3.7)

End

Update the trail values according to Eq. (3.6)

Update the transition probability according to Eq. (3.4)

Step 4 Next search

Set NC = NC+1

For every combination (i,j)

$\Delta\tau_{ij} = 0$

End

Step 5 Termination

If (NC < NC_{max})

Then

Go to step 2

Else

Print the best feasible solution

Stop

End

End

3.4.2 Components of Ant Colony Optimization

3.4.2.1 State transition rule

The state transition rule of the ant colony is given in Eq. (3.3). This equation represents the probability that ant k selects a link connecting node i and node j :

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{m=1}^{M_i}\left[\tau_{im}(t)\right]^{\alpha}\left[\eta_{im}(t)\right]^{\beta}} \quad (3.3)$$

where m represents number of ant

M represents total number of ant

k represents ant number

τ represents the pheromone intensity

η represents the heuristic information between node i and node j

α represents the relative importance of the trail

β represents the relative importance of the heuristic information

The specific heuristic information of the problem is:

$$\eta_{ij} = \frac{1}{C_{ij}} \quad (3.4)$$

where C represents the associated cost

Therefore, the level of link with smaller cost has greater probability to be chosen.

3.4.2.2 Global updating rule

During the solution construction, it is no guarantee that an ant will construct a feasible solution, which obeys the reliability constraint. The pheromone updating treats the unfeasible solution. The amount of pheromone, deposited by ants, is set to a high value if the generated solution is feasible. On the other hand, this value is set to a low value if it is infeasible. Therefore, this value depends on the solution quality. Infeasibility can be handled by assigning the penalty which is in proportion to the amount of reliability violations. In case of a feasible solution, an additional penalty is introduced to improve its quality.

Following the above remarks, the trail intensity is updated as follows:

$$\tau_{ij}(new) = \rho\tau_{ij}(old) + \Delta\tau_{ij} \tag{3.5}$$

where ρ represents a coefficient

$(1-\rho)$ represents the evaporation of trail

$\Delta\tau$ represents the deposited pheromone, is given by:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3.6}$$

where m represents the number of ant

$\Delta\tau_{ij}^{k}$ represents the deposited pheromone of ant k , is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q \cdot \text{penalty}_k \cdot C_k & \text{if the } k^{\text{th}} \text{ ant chooses level } j \text{ for link } i \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where Q represents a positive number

penalty_k is defined as follows:

$$\text{penalty}_k = \left(\frac{C_k}{C^*}\right)^{a} \tag{3.8}$$

where C_k represents the cost obtained by the k^{th} ant

C^* represents the best obtained solution

a represents the relative importance of penalties

3.5 Improved Ant Colony Optimization Algorithm (IACO)

3.5.1 Principle of Improved Ant Colony Optimization

A major weakness of the conventional ACO algorithm is stagnation that all ants take the same position. If this situation occurs, the algorithm may be trapped in a local optimal point. To alleviate the stagnation problem of conventional ACO algorithms, two improvement procedures are applied in order to improve the Ant Colony Optimization method to guarantee the diversity of ants. This approach is called Improved Ant Colony Optimization (IACO). The additional procedures are specific improvement algorithms called ***improvement and*** *neighborhood search* and *re-initialization*.

3.5.2 Additional algorithms of Improved Ant Colony Optimization

3.5.2.1 Improvement and neighborhood search

The general step of an iterative procedure consists in constructing from a current solution *i* a next solution *j* and in checking whether one should stop there or perform another step. Neighbourhood search methods are iterative procedures in which a neighbourhood *N(i)* is defined for each feasible solution *i*, and the next solution *j* is searched among the solutions in *N(i)*. The neighborhood search method which has been used for finding an approximation to the minimum value of a function *f* on a set *S* is the descent method which is described below:

Descent method

Step 1: Choose an initial solution *i* in *S*

Step 2: Find a best *j* in *N(i)* (i.e. such that *f(j)* £ *f(k)* for any *k* in *N(i)*)

Step 3: If *f(j)* ³ *f(i)* then stop. Else set *i=j* and go to Step 2

3.5.2.2 Re-initialization

During searching, the search process gets the repeated solution for a long time. That means the process cannot find a better solution or escapes from this solution. Therefore, this solution can be either a local or a global solution. In case of the global solution, the search process will be stopped as it has been gotten the best solution. On the other hand, if the global solution is unknown, the algorithm will assume that this solution is a local solution. The process is stroked on a local solution for a long time and the re-initialization process will be applied to reset parameter values of *state transition rule* and *global updating rule*. This mechanism helps the process to continue searching and find better solutions.

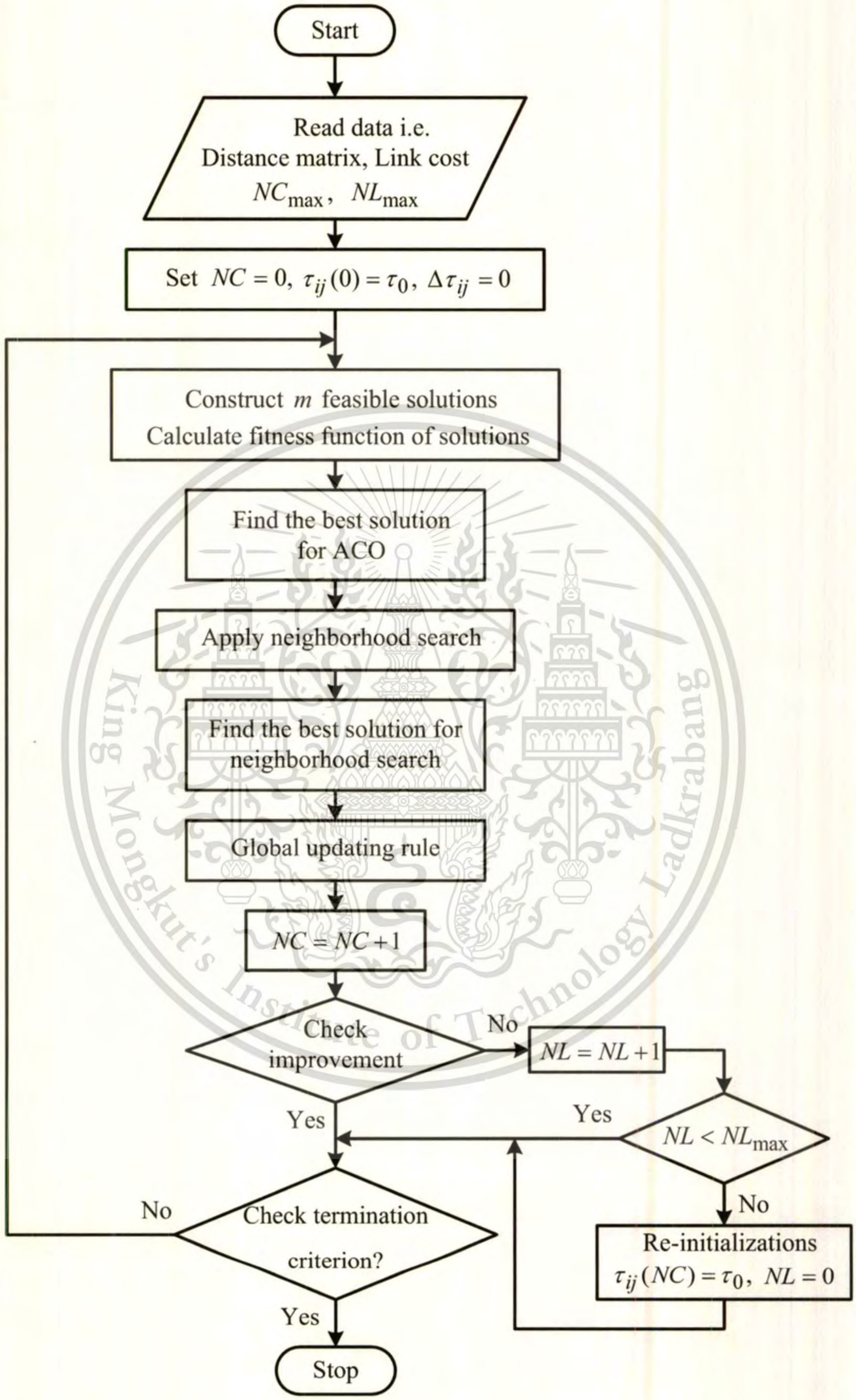


Figure 3.12 A flow chart of an Improved Ant Colony Optimization algorithm

The detail of an IACO algorithm is described as the following steps and presented in Figure 3.12:

Step 1 Initialization

Set NC = 0 /* NC: cycle counter */

For every combination (i,j)

Set an initial value $\tau_{ij}(0) = \tau_0$ and $\Delta\tau_{ij} = 0$

End

Step 2 Construct feasible solutions

For k=1 to m /* m: number of ants */

For i=1 to n /* n: number of links */

Choose a level of connection with transition probability given by Eq. (3.3)

End

Calculate fitness function f_k /* f_k: fitness value of each ant */

End

Update the best solution

Step 3 Apply the neighborhood search

For k=1 to m

For i=1 to (2*n)

If i = odd

Change the chosen level of link i with level p by level p+1

Else

Change the chosen level of link i with level p by level p-1

End

Calculate fitness function f_k

If $f_k \geq f_o$

Accept for exchanging

Record the obtained solution

Else

Not accept for exchanging

End

End

End

Update the best solution

Step 4 Global updating rule

For every combination (i,j)

For k=1 to m

Find $\Delta\tau_{ij}^{k}$ according to Eq. (3.7)

End

Update $\Delta\tau_{ij}$ according to Eq. (3.6)

End

Update the trail values according to Eq. (3.5)

Update the transition probability according to Eq. (3.3)

Step 5 Next search

Set NC = NC+1

For every combination (i,j)

$\Delta\tau_{ij} = 0$

End

Step 6 Re-initializations

If the best solution has not been improved for a long time

Then

Set an initial value $\tau_{ij}(0) = \tau_0$

End

Step 7 Termination

If ($NC < NC_{max}$)

Then

Go to step 2

Else

Print the best feasible solution

Stop

End

End

3.6 Numerical Optimization Tests

In order to evaluate the performance of the IACO algorithm, some classical benchmark functions [23] were employed. Results of the IACO algorithm have been compared with GA, TSA, PSO and ACO.

3.4.1 Numeric Function Optimization

Five well-known minimization test functions, parameter ranges, formulations and global optimal values of these functions are given in Table 3.1. In the experiments, $f_1(x)$ Schaffer function has 2 parameters, while $f_2(x)$ Sphere function and $f_3(x)$ Rosenbrock have 5 parameters. $f_4(x)$ Griewank function and $f_5(x)$ Rastrigin's function have 10 parameters.

Function $f_1(x)$ is a two-dimension Schaffer function. x_i is in the interval of $[-10, 10]$. The global minimum value for this function is 0 and the optimum solution is $x_i = (x_1, x_2) = (0,0)$. Surface plot and contour lines of $f_1(x)$ are presented in Figure 3.13.

Function $f_2(x)$ is a Sphere function, which is continuous, convex and unimodal. x_i is in the interval of $[-100,100]$. The global minimum value for this function is 0 and the optimum solution is $x_i = (x_1, x_2, x_3, x_4, x_5) = (0,0,0,0,0)$. Surface plot and contour lines of $f_2(x)$ are presented in Figure 3.14.

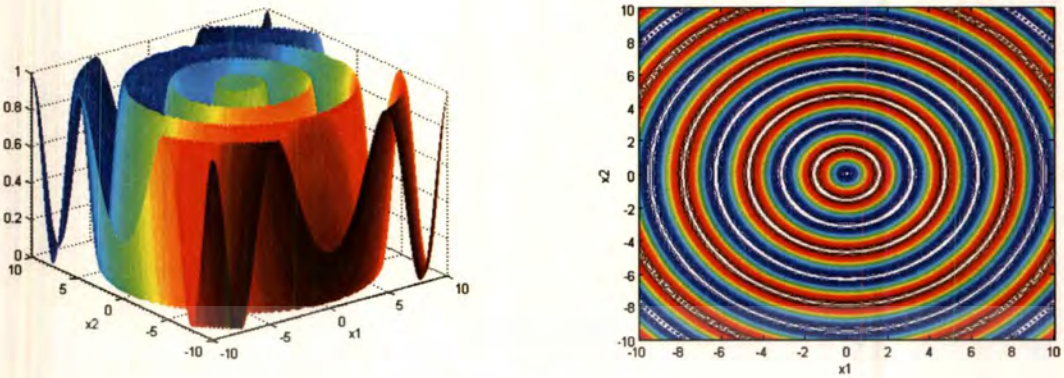
Function $f_3(x)$ is a well-known classic optimization problem: Rosenbrock valley. The global optimum is inside a long, narrow, parabolic-shaped flat valley. Since it is difficult to converge to the global optimum of this function, the variables are strongly dependent, and the gradients generally do not point towards the optimum, this problem is repeatedly used to test the performance of the optimization algorithms. x_i is in the interval of $[-2.048, 2.048]$. The global minimum value for this function is 0 and the optimum solution is $x_i = (x_1, x_2, x_3, x_4, x_5) = (1,1,1,1,1)$ .Global optimum is the only optimum, and the function is unimodal. Surface plot and contour lines of $f_3(x)$ are presented in Figure 3.15.

Function $f_4(x)$ is a Griewank function, which is also a non-linear and multimodal function. The terms of the summation produce a parabola, while the local optima are above parabola level. The dimensions of the search range increase on the basis of the product. x_i is in the interval of $[-600, 600]$. The global minimum value for this function is 0 and the corresponding global optimum solution is $x_i = (x_1, x_2, \ldots, x_{10}) = (100, 100, \ldots, 100)$. Since the number of local optima increases with the dimensionality, this function is strongly multimodal. Surface plot and contour lines of $f_4(x)$ are presented in Figure 3.16.

Function $f_5(x)$ is a Rastrigin's function, which is a difficult problem due to the large search space and large number of local minima. This function contains millions of local optima in the interval of consideration. The difficult part about finding optimal solutions to this function is that an optimization algorithm is easily trapped in a local optimum on its way towards the global optimum. x_i is in the interval of $[-5.12, 5.12]$. The global minimum value for this function is 0 and the optimum solution is $x_i = (x_1, x_2, \ldots, x_{10}) = (0, 0, \ldots, 0)$. Surface plot and contour lines of $f_5(x)$ are presented in Figure 3.17.

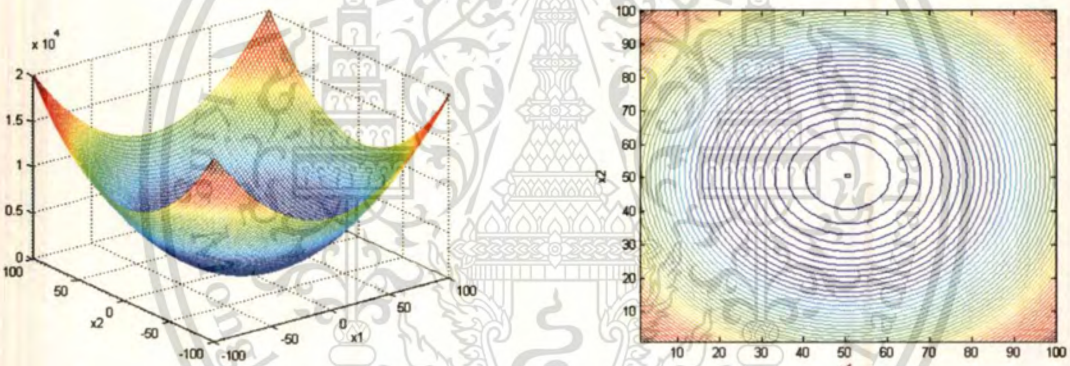
Table 3.1 Numerical benchmark functions

Function	Range	Solution
$f_1(x) = 0.5 + \dfrac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	$-10 \le x_i \le 10$	$f_1(0) = 0$
$f_2(x) = \sum_{i=1}^{n} x_i^2$	$-100 \le x_i \le 100$	$f_2(0) = 0$
$f_3(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$-2.048 \le x_i \le 2.048$	$f_3(1) = 0$
$f_4(x) = \sum_{i=1}^{n} \dfrac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\dfrac{x_i}{\sqrt{i}}\right) + 1$	$-600 \le x_i \le 600$	$f_4(0) = 0$
$f_5(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$-5.12 \le x_i \le 5.12$	$f_5(0) = 0$



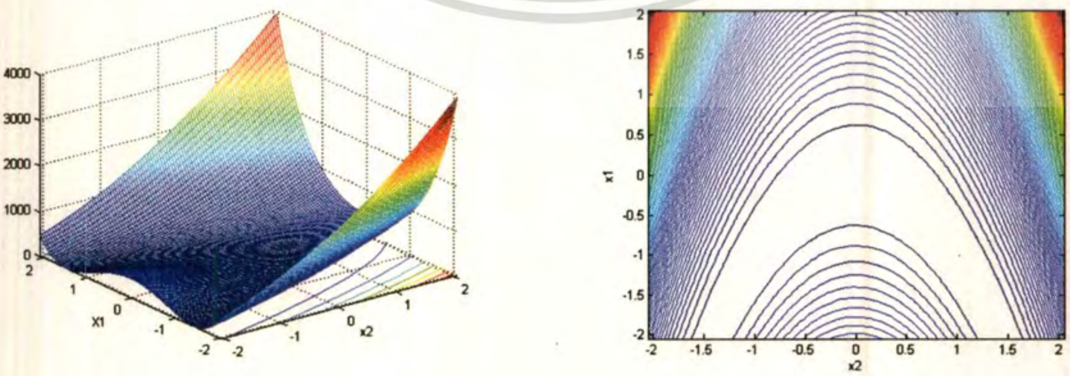
(a) (b)

Figure 3.13 Schaffer function; **(a)** surface plot, and **(b)** contour lines



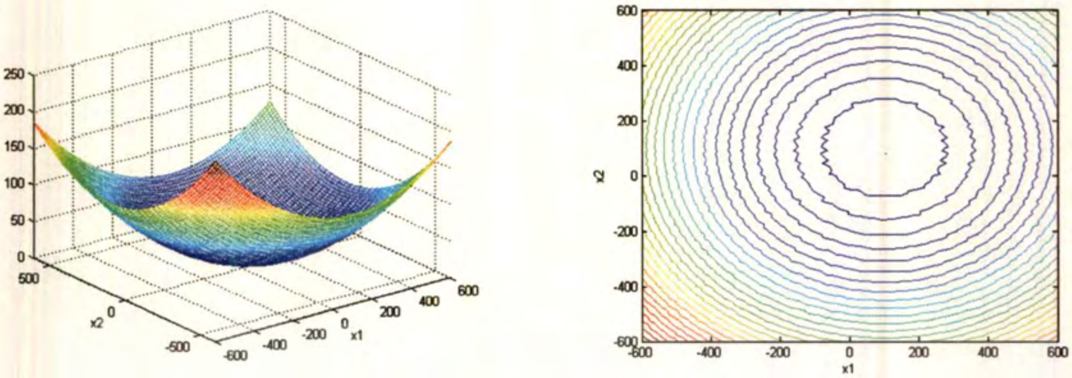
(a) (b)

Figure 3.14 Sphere function; **(a)** surface plot, and **(b)** contour lines



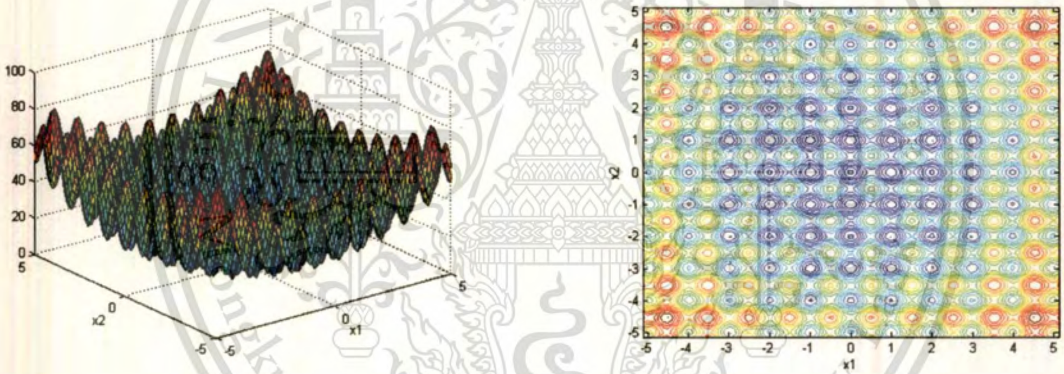
(a) (b)

Figure 3.15 Rosenbrock function; **(a)** surface plot, and **(b)** contour lines



(a) (b)

Figure 3.16 Griewank function; **(a)** surface plot and, **(b)** contour lines



(a) (b)

Figure 3.17 Rastrigin's function; **(a)** surface plot and, **(b)** contour lines

3.4.2 Simulation and Results

The statistic results are derived from 30 trials, with 2-minute maximum time each, such as the minimum, maximum and average costs, standard deviation, computational time and percentage of approaching optimal solution for these functions, as presented in Table 3.2 - 3.6.

As seen from the results that are presented in the tables, the proposed IACO algorithm produces the best performance in terms of cost, computational time and percentage of approaching optimal solution.

Table 3.2 The results obtained by GA, TSA, PSO, ACO and IACO for solving a Schaffer function

Algorithm	Max Cost	Average Cost	Min Cost	Standard Deviation	CPU Time (s)	% Get optimal
GA	4.855787E-07	2.588906E-07	5.035579E-08	1.284828E-07	48.38	100
TSA	9.990988E-07	4.999105E-07	3.517738E-08	3.057724E-07	66.02	100
PSO	1.965572E-07	9.980305E-08	4.889027E-08	4.225685E-08	6.56	100
ACO	7.789202E-07	3.943427E-07	5.309629E-08	2.290206E-07	10.52	100
IACO	7.969155E-08	3.616158E-08	1.149785E-09	2.386150E-08	1.57	100

Table 3.3 The results obtained by GA, TSA, PSO, ACO and IACO for solving a Sphere function

Algorithm	Max Cost	Average Cost	Min Cost	Standard Deviation	CPU Time (s)	% Get optimal
GA	3.656064E-07	1.479403E-07	4.120297E-09	1.061122E-07	70.23	100
TSA	9.954358E-08	4.571486E-08	2.023325E-09	2.967943E-08	73.44	100
PSO	3.963821E-08	2.780595E-08	4.018806E-09	9.347978E-09	4.45	100
ACO	8.865825E-08	4.789913E-08	8.478699E-09	2.771570E-08	9.91	100
IACO	3.595975E-09	2.072073E-09	4.704794E-10	8.802275E-10	1.71	100

Table 3.4 The results obtained by GA, TSA, PSO, ACO and IACO for solving a Rosenbrock function

Algorithm	Max Cost	Average Cost	Min Cost	Standard Deviation	CPU Time (s)	% Get optimal
GA	8.462722E-07	4.158122E-07	1.162158E-07	2.350917E-07	71.56	100
TSA	9.362512E-07	5.452672E-07	1.168046E-07	2.398151E-07	81.91	100
PSO	8.975427E-08	5.290316E-08	1.006107E-08	2.400884E-08	7.33	100
ACO	2.976119E-07	2.013075E-07	8.399906E-08	7.183145E-08	19.45	100
IACO	6.916146E-08	3.436999E-08	5.642636E-09	1.699658E-08	2.73	100

Table 3.5 The results obtained by GA, TSA, PSO, ACO and IACO for solving a Griewank function

Algorithm	Max Cost	Average Cost	Min Cost	Standard Deviation	CPU Time (s)	% Get optimal
GA	1.648353E-05	9.457299E-06	1.563919E-06	4.483127E-06	76.29	56
TSA	1.699132E-05	8.823626E-06	7.727716E-07	4.857630E-06	68.79	60
PSO	1.411494E-05	5.344576E-06	4.189883E-07	3.652975E-06	44.57	86
ACO	1.960441E-05	7.155685E-06	6.646413E-07	5.148640E-06	59.65	73
IACO	5.986868E-06	3.875797E-06	6.037241E-07	1.506074E-06	4.65	100

Table 3.6 The results obtained by GA, TSA, PSO, ACO and IACO for solving a Rastrigin's function

Algorithm	Max Cost	Average Cost	Min Cost	Standard Deviation	CPU Time (s)	% Get optimal
GA	1.437266E-05	7.971044E-06	8.068149E-07	4.591210E-06	77.24	63
TSA	1.465138E-05	7.911400E-06	9.253190E-07	4.557194E-06	84.41	66
PSO	7.994649E-06	4.282686E-06	6.655791E-08	2.451279E-06	24.50	100
ACO	1.467910E-05	6.855292E-06	5.602892E-07	4.070891E-06	52.96	80
IACO	3.966981E-06	2.119745E-06	2.135902E-08	1.170877E-06	8.46	100

CHAPTER 4

Network Reliability Optimization

4.1 Reliability Calculation

The problem of calculating or estimating the reliability of a network is one of the research areas related to the economic network design. There are two main approaches of finding the network reliability. The first one is the exact calculation through analytic methods [24, 25] and the second one is the estimation calculation through Monte Carlo simulation [26, 27]. For the all-terminal network reliability problem, it is difficult to find the exact reliability because these methods generally lose efficiency when a network approaches a fully connected state. There are also the upper and lower bound expressions for network reliability [28], however, they lose the effective surrogates in all-terminal design process. Furthermore, many bounds of procedures and improved efficiency simulations depend on the assumption that all links have the same reliability, which is related to this research. In this thesis, the network reliability was calculated (or estimated) using a backtracking procedure (or a classic Monte Carlo procedure). Due to the computational tractable size, a backtracking algorithm is used to correctly calculate the system reliability, $R(x)$. The outline of the backtracking algorithm is given as follows:

Step 0*: *Initialization

Mark all links as free; create a stack, which is initially empty.

Step 1*: *Generate modified cut-set

(a) Find a set of free links that together with all inoperative links will form a network-cut.

(b) Mark all the links found in 1(a) inoperative and add them to the stack.

(c) The stack now represents a modified cut-set; add its probability into a cumulative sum.

Step 2*: *Backtracking

(a) If the stack is empty, end.

(b) Take a link off the top of the stack.

(c) If the link is inoperative and it is operative a spanning tree of operative links exists, then mark it free and go to 2(a).

(d) If the link is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack, and go to Step 1.

(e) If the link is operative, then mark it free and go to 2(a).

As mentioned before, the algorithm above is for all-terminal reliability and needs to be modified for use in a source-destination design problem as given below:

Step 0*: *Initialization

Mark all links as free; create a stack, which is initially empty

Step 1*: *Generate modified cut-set

(a) Find a set of free links that together with all inoperative links will form a source-destination cut.

(b) Mark all the links found in 1(a) inoperative and add them to the stack.

(c) The stack now represents a modified cut-set; add its probability to a cumulative sum.

Step 2*: *Backtracking

(a) If the stack is empty, end.

(b) Take a link off the top of the stack.

(c) If the link is inoperative and if made operative, a path from the source to the destination exists, then mark it free and go to 2(a).

(d) If the link is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack, and go to Step 1.

(e) If the link is operative, then mark it free and go to 2(a).

For larger networks, Monte Carlo simulation is used to estimate network reliability. The network is simulated t times with the design and the link reliabilities that are given:

Initialize $i = 0$, $c = 0$

***Step 0*:** While $i < t$ Repeat

***Step 1*:** Randomly generate network

(a) $i = i + 1$

***Step 2*:** Check if the network forms a spanning tree.

(a) If the network forms a spanning tree, then $c = c + 1$, go to Step 0.

(b) If the network does not form a spanning tree, go to Step 0.

Step 3: $R(x) = c / t$

When the need of a network's reliability simulation has arisen, two issues become important. The first one is the biased estimator. The other is the variance of estimation. Every referenced Monte Carlo technique is an unbiased estimator where the variance of the Monte Carlo method described above is:

$$\mathrm{Var}(R(x)) = \frac{R(x)(1-R(x))}{t} \tag{4.1}$$

To get more accurate reliability estimation, t should have a larger number.

4.2 Application of IACO Algorithm for Network Topology Design

For an application of an IACO algorithm to the design of a network topology, it is convenient to represent the network by a graph $G=(N,E)$, where N are the nodes and E is the set of links. Ants' cooperation uses the indirect form of communication mediated by pheromone that they deposit on the links of the graph G while building solutions.

For example, an all-terminal network has 5 nodes and 10 links as shown in Figure 4.1. Each link can choose 4 connection levels. This network can model as the routes between the nest and food source for IACO as shown in Figure 4.2. This model reveals that the topology of a network can be constructed from a selection of the connection level of each link, which seems to be the ant's route between the nest and food source. Solutions of m ants and pheromone vector of connection levels, k = 4 (0, 1, 2, 3) have been formulated as $n \times n \times m$ matrix which is shown in Figure 4.3 and $n \times n \times 4$ matrix which is shown in Figure 4.4, respectively. In additional, there are 2 termination conditions applied in all simulations which are optimum solution and maximum time.

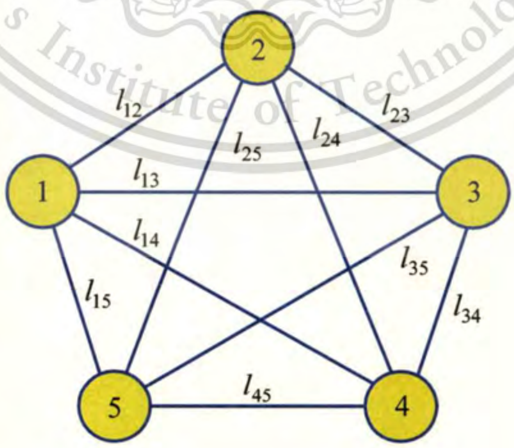


Figure 4.1 A five-node communication network system

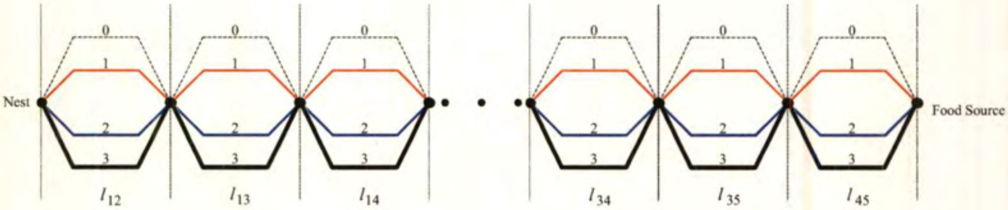


Figure 4.2 A network model as the routes of ants between nest and food source

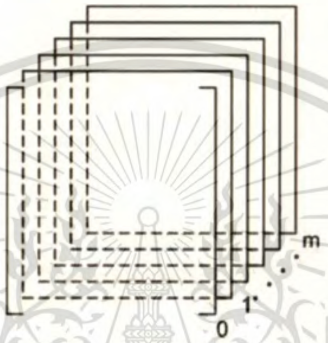


Figure 4.3 $n \times n \times m$ matrix of solution x of m ants

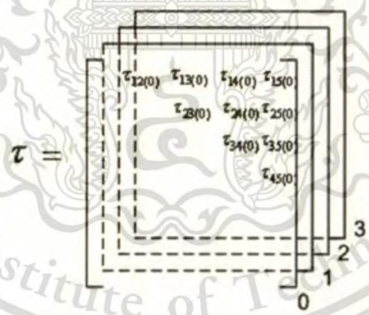


Figure 4.4 $n \times n \times 4$ matrix of pheromone value of 4 connection levels

As mentioned in the previous chapter, the neighborhood search and re-initializations in the Improved Ant Colony Optimization (IACO) algorithm to improve the efficiency of an ACO algorithm are proposed.

The neighborhood search algorithm is shown in step 3 of the IACO algorithm, and it proceeds to change in turn each connection level of chosen links by another connection level. For each link, connection levels are indexed in ascending order in accordance with their reliability. A solution $S = \{u, v, ...\}$ indicates that link 1 uses a connection level with index u, link 2 uses a connection level with index v, etc.

Let us consider, for example, a source-destination network that has 4 nodes and 5 links. Each link can choose 4 connection levels as shown in Fig. 2. Suppose that the obtained solution at a given cycle is $S = \{1,3,2,0,1\}$. The local search has to evaluate the following solutions:

$$S = \{2,3,2,0,1\}\ S = \{0,3,2,0,1\}, S = \{1,3^*,2,0,1\} S = \{1,2,2,0,1\},$$
$$S = \{1,3,3,0,1\}\ S = \{1,3,1,0,1\}, S = \{1,3,2,1,1\} S = \{1,3,2,0^*,1\},$$
$$S = \{1,3,2,0,2\}\ S = \{1,3,2,0,0\}$$

* The same result because this connection level is a maximum or minimum index.

4.3 Test Problems

The mathematical formulation of test problems to minimize cost has been presented by Eq. (2.3). In each test problem, there are 6 reliability constraints which are 0.850, 0.900, 0.950, 0.990, 0.995 and 0.999.

To assess the feasibility of the Improved Ant Colony Optimization (IACO), three different test problems are applied and compare its results with Genetic Algorithm (GA), Tabu Search Algorithm (TSA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO). The mentioned optimization methods were implemented in MATLAB® package and the simulation cases done on a Intel® Core2 Duo 1.66 GHz personal microcomputer with 1-GB RAM under Windows XP Operating System. Each studied system was run 30 times with differential random initial solutions. In order to evaluate the performance of each technique; maximum, minimum, average and standard deviation of the generation costs as well as average of computational time to get near an optimum solution are used for evaluations.

4.3.1 Computer Programming Parameters

Parameters of 5 methods used in computer programming have been shown below;

Genetic Algorithm parameters

- Population size = 200
- Generations = 200
- Binary bits = 12
- Crossover rate = 0.8
- Mute rate = 0.01

- Crossover parameter = 0.5

Particle Swarm Optimization parameter

- Population size = 200
- Generations = 200
- Inertia weight factor $\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{iter_{\max}} \times iter$, where $\omega_{\max} = 0.9$ and $\omega_{\min} = 0.4$
- The limit of change in velocity of each member in an individual was as $V_{P_d}^{\max} = 0.5P_d^{\max}$, $V_{P_d}^{\min} = -0.5P_d^{\min}$
- Acceleration constant $c_1 = 1.4$ and $c_2 = 1.4$

Tabu Search parameters

- Tabu list size = 2*n
- Restriction period = 10
- Frequency limit = 7

Ant Colony Optimization parameter

- Number of ant = 10
- Evaporation value = 0.05
- Relative value of the trail (α) = 1
- Relative value of heuristic information (β) = 0.8

Improved Ant Colony Optimization parameter

- Number of ant = 10
- Evaporation value = 0.05
- Relative value of the trail (α) = 1
- Relative value of heuristic information (β) = 0.8

4.3.2 Stopping criterion

The search algorithm will be terminated following some stopping rules. The rules may be a fixed number of iterations, a fixed number of CPU time, or a fixed number of consecutive iterations without an improvement in the best objective function value, etc

In this thesis, there are 2 immediate stopping conditions which are the following items:

- The accuracy of the best solution is lower than the expected value
- The maximum allowable computational time is reached

4.3.3 Test Problem 1: Five-Node Network

A communication network has five nodes and ten links as shown in Figure 4.1. This problem is referred to *Jan et, al* [1], which is expanded by changing the status of links from a simple on/off state to four possible statuses. The link cost relates with distances between a pair of nodes, and the unit costs that depend on the reliabilities of connection, as shown in Table 4.1. Each link can select 4 choices of connection levels, $k = 4$, so its search space size is $4^{(5\times4)/2} = 1{,}048{,}576$. This problem is considered at six different system reliability constraints, and the backtracking algorithm is employed to calculate $R(x)$ exactly. The distance matrix of links is given as follows:

$$[l_{ij}] = \begin{bmatrix} - & 32 & 54 & 62 & 25 \\ & - & 34 & 58 & 45 \\ & & - & 36 & 52 \\ & & & - & 29 \\ & & & & - \end{bmatrix}$$

Table 4.1 Link unit costs and corresponding reliabilities

Connection Type	**Reliability**	**Cost/Unit Distance($/km)**
0 (not connected)	0.00	0
1	0.85	8
2	0.90	10
3	0.95	14

4.3.4 Test Problem 2: Source-Destination Network

This problem shows that the flexibility of the approach by considering economic design of 18-link of a source-destination network, as shown in Figure 4.5. This problem is taken from the literature [5] and has 6.9×10^{10} possible architectures, thus precluding enumeration to identify the optimal design. The distance matrix for this problem appears in Table 4.2. This problem is considered at six different system reliability constraints and uses the backtracking algorithm to calculate the exact value of $R(x)$.

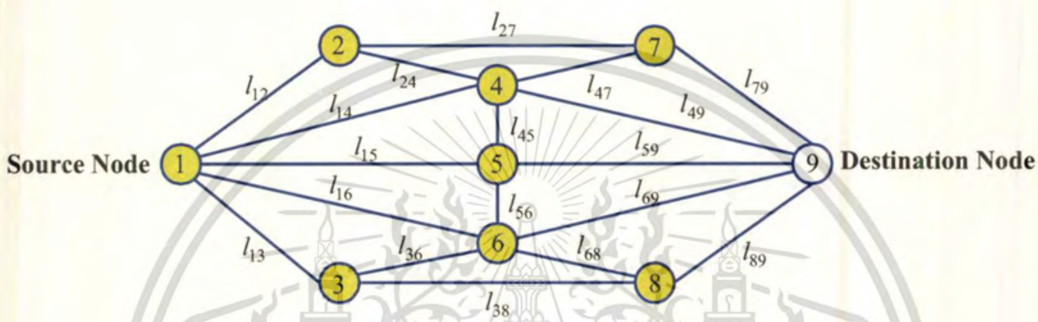


Figure 4.5 A source-destination network

Table 4.2 Distance matrix for source-destination network

Node no.	2	3	4	5	6	7	8	9
1	58	63	60	63	58	-	-	-
2	-	42	-	-	-	60	-	-
3	-	-	20	-	-	42	-	63
4	-	-	-	20	-	-	-	60
5	-	-	-	-	42	-	42	63
6	-	-	-	-	-	-	60	-
7	-	-	-	-	-	-	-	58
8	-	-	-	-	-	-	-	58

4.3.5 Test Problem 3: 19 Districts in Bangkok, Thailand

To demonstrate the applicability of this work on a realistic application, the network of 19 districts in Bangkok is therefore applied for an example of a scaled-up problem. This 19-node network problem was constructed by selecting districts in Bangkok, as shown in Figure 4.6 and computing the Euclidean distances between them using their coordinates, as shown in Table 4.3 and using the four-link choices from Table 4.1. The search space of this problem is 8.959×10^{102}. Besides the scale-up issue, the difference between this problem and the other design problems is that the former illustrates the flexibility of the IACO by reversing the constraint and the objective function. This problem is considered at six different system reliability constraints. In this case, the Monte Carlo simulation is used to estimate the network reliability.

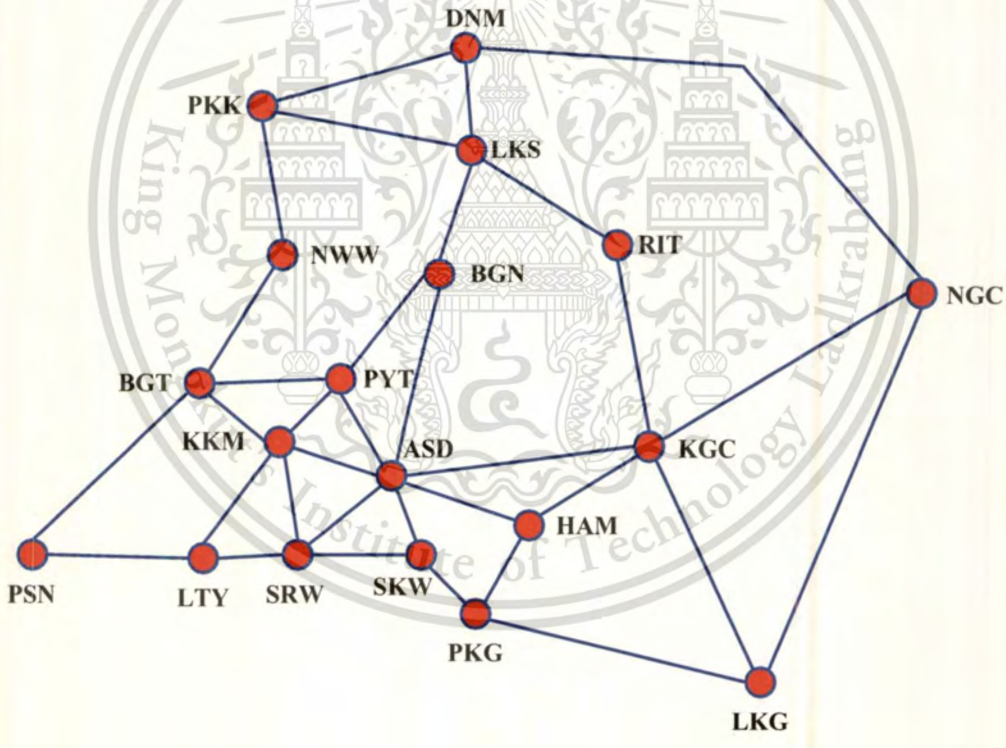


Figure 4.6 A network of 19 districts in Bangkok, Thailand

Table 4.3 Distance matrix of 19 districts in Bangkok, Thailand

	LKS	PKK	NWW	RIT	BGN	NGC	BGT	PYT	KKM	KGC	ASD	PSN	LTY	SRW	SKW	HAM	PKG	LKG
DNM	8.10	19.40	27.88	17.50	12.80	45.41	29.92	20.30	30.20	25.90	36.59	48.62	36.61	38.90	40.47	31.90	38.00	66.06
LKS	-	11.30	19.78	9.40	4.70	45.28	21.82	12.20	22.10	17.80	28.49	40.52	28.51	30.80	32.37	23.80	29.90	41.75
PKK	-	-	8.48	20.70	16.00	62.01	25.80	25.58	26.08	29.10	39.79	44.50	32.49	32.18	43.67	45.49	49.37	49.75
NWW	-	-	-	29.18	24.48	70.49	17.32	17.10	17.60	19.35	25.10	36.02	24.01	23.70	28.98	30.80	34.68	54.27
RIT	-	-	-	-	14.10	35.88	31.22	21.60	31.50	8.40	20.10	49.92	37.91	30.00	26.00	14.20	20.30	32.35
BGN	-	-	-	-	-	52.96	17.12	7.50	17.40	25.48	23.79	35.82	23.81	34.69	27.67	29.49	33.37	52.96
NGC	-	-	-	-	-	-	56.80	47.18	57.08	27.48	39.18	65.74	55.88	50.08	43.06	33.48	39.58	20.65
BGT	-	-	-	-	-	-	-	9.62	5.32	28.97	17.62	18.70	11.96	17.76	12.92	23.32	18.62	38.21
PYT	-	-	-	-	-	-	-	-	9.90	19.35	8.00	28.32	16.31	16.00	11.88	13.70	17.58	37.17
KKM	-	-	-	-	-	-	-	-	-	20.90	9.20	16.27	6.41	6.10	7.60	14.90	13.30	32.89
KGC	-	-	-	-	-	-	-	-	-	-	11.70	38.26	28.40	22.60	12.10	6.00	12.10	23.95
ASD	-	-	-	-	-	-	-	-	-	-	-	26.56	16.70	10.90	3.88	5.70	9.58	29.17
PSN	-	-	-	-	-	-	-	-	-	-	-	-	9.86	15.66	24.96	36.76	30.66	50.25
LTY	-	-	-	-	-	-	-	-	-	-	-	-	-	5.80	15.10	26.90	20.80	40.39
SRW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9.30	21.10	15.00	34.59
SKW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	11.80	5.70	25.29
HAM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6.10	25.69
PKG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	19.59

DNM: Donmuang LKS: Laksi PKK: Pakkret

NWW: Ngamwongwan RIT: Ram-intra BGN: Bangken

NGC: Nongchok BGT: Bangplad PYT: Paholyothin

KKM: Krungkasem KGC: Klongchan ASD: Asok Dindeang

PSN: Pasicharoen LTY: Latya SRW: Surawong

SKM: Sukhumwit HAM: Huamark PKG: Phrakanong

LKG: Ladkrabang

4.4 Simulation Results and Comparisons

Five methods, which are GA, TSA, PSO, ACO and IACO, are employed to examine 3 test problems with six different system reliability constraints.

4.4.1 Comparison in Term of Optimum Solution

To show performance in term of finding optimum solution, the following conditions have been applied;

- **Example 1;** the 60-second maximum time has been used as the stopping criteria where the optimum solutions, which are 1724, 1904, 2184, 3638, 4250 and 5498 USD subjected to 0.8500, 0.900, 0.950, 0.990, 0.995 and 0.999 reliability constraints, respectively.
- **Example 2;** the120-second maximum time has been used as the stopping criteria where the optimum solutions, which are 1990, 2328, 2826, 4074, 4464 and 6008 USD subjected to 0.8500, 0.900, 0.950, 0.990, 0.995 and 0.999 reliability constraints, respectively.
- **Example 3;** the 900-second maximum time has been used as the stopping criteria where the optimum solutions, which are 96706, 117696, 137140, 189820, 252837 and 295627 USD subjected to 0.8500, 0.900, 0.950, 0.990, 0.995 and 0.999 reliability constraints, respectively.

In order to demonstrate the efficiency of the proposed IACO method, the distribution outlines of the best solution of each trial are considered. Results have been shown in Table 4.4 – 4.6. Figure 4.7 - 4.24 demonstrate the distribution outlines of the best solution of each trial for 6 reliability levels in test problem 1, 2 and 3, respectively. Most of the costs obtained by the proposed IACO method are lower than those of the compared methods and, thus the efficiency of the proposed IACO method is superior to other methods.

Table 4.4 Summary results of a five-node network problem (performed 60 seconds)

Reliability	Optimal cost ($)	Configuration	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	% Get optimal	CPU Time (s)
0.999	**5498**	**3323323333**	**GA**	5498	5498	5498	0.00000	100	12.49
			TSA	5498	5498	5498	0.00000	100	9.93
			PSO	5530	5498	5498	0.00000	100	5.25
			ACO	5530	5498	5498	0.00000	100	7.11
			IACO	5498	5498	5498	0.00000	100	4.40
0.995	**4250**	**3303222313**	**GA**	4250	4250	4250	0.00000	100	47.21
			TSA	4250	4250	4250	0.00000	100	43.65
			PSO	4250	4250	4250	0.00000	100	32.92
			ACO	4250	4250	4250	0.00000	100	35.50
			IACO	4250	4250	4250	0.00000	100	20.74
0.990	**3638**	**3203312303**	**GA**	3900	3760	3638	90.96168	13	57.12
			TSA	3886	3700	3638	57.59873	17	55.53
			PSO	3876	3694	3638	86.66676	57	39.77
			ACO	3686	3654	3638	18.78725	50	44.89
			IACO	3638	3638	3638	0.00000	100	23.35
0.950	**2184**	**3003300303**	**GA**	2692	2252	2184	121.63125	56	57.35
			TSA	2622	2223	2184	98.47133	80	31.52
			PSO	2184	2184	2184	0.00000	100	12.31
			ACO	2444	2193	2184	47.46929	97	22.11
			IACO	2184	2184	2184	0.00000	100	10.24
0.900	**1904**	**3003200203**	**GA**	2402	1982	1904	115.77198	43	47.55
			TSA	2124	1919	1904	55.81579	93	17.51
			PSO	1904	1904	1904	0.00000	100	14.35
			ACO	1904	1904	1904	0.00000	100	14.27
			IACO	1904	1904	1904	0.00000	100	4.70
0.850	**1724**	**2003300102**	**GA**	1764	1729	1724	10.06188	90	21.11
			TSA	1724	1724	1724	0.00000	100	15.63
			PSO	1724	1724	1724	0.00000	100	11.85
			ACO	1724	1724	1724	0.00000	100	8.11
			IACO	1724	1724	1724	0.00000	100	2.09

Table 4.5 Summary results of a source-destination network problem (performed 120 seconds)

Reliability	Optimal cost ($)	Configuration	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	% Get optimal	CPU Time (s)
0.999	6008	033300000001013330	GA	6008	6008	6008	0.000000	100	63.41
			TSA	6008	6008	6008	0.000000	100	55.48
			PSO	6008	6008	6008	0.000000	100	21.60
			ACO	6008	6008	6008	0.000000	100	26.42
			IACO	6008	6008	6008	0.000000	100	18.93
0.995	4464	022200000000003330	GA	5382	4795	4464	272.423571	13	112.42
			TSA	4970	4681	4464	193.439717	36	105.20
			PSO	4640	4495	4464	59.634134	76	40.04
			ACO	4596	4506	4464	57.256069	63	50.87
			IACO	4464	4464	4464	0.000000	100	13.34
0.990	4074	013100000000001330	GA	5212	4190	4074	267.066792	76	72.66
			TSA	4952	4189	4074	246.057794	80	69.75
			PSO	4074	4074	4074	0.000000	100	37.57
			ACO	4074	4074	4074	0.000000	100	42.27
			IACO	4074	4074	4074	0.000000	100	22.39
0.950	2826	002100000000000330	GA	3446	2980	2826	145.287096	30	106.43
			TSA	3112	2922	2826	110.599724	53	93.21
			PSO	3016	2850	2826	59.211718	83	51.11
			ACO	3248	2900	2826	129.451811	70	72.68
			IACO	2826	2826	2826	0.000000	100	36.18
0.900	2328	011000000000001300	GA	2648	2386	2328	83.874263	57	90.23
			TSA	2530	2374	2328	64.824263	60	79.92
			PSO	2394	2332	2328	16.744737	93	38.78
			ACO	2394	2337	2328	22.819230	87	49.98
			IACO	2328	2328	2328	0.000000	100	32.54
0.850	1990	002000001000000220	GA	2312	2074	1990	100.527551	50	65.98
			TSA	2335	2104	1990	141.804463	57	63.02
			PSO	1990	1990	1990	0.000000	100	46.86
			ACO	2180	2003	1990	48.204545	93	61.39
			IACO	1990	1990	1990	0.000000	100	43.48

Table 4.6 Summary results of a19 districts in Bangkok, Thailand (performed 900 seconds)

Reliability	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	Near Optimal cost ($)	% Get near optimal	CPU Time (s)
0.999	GA	344055	326513	302649	11134.82	< 310408	10	848.41
	TSA	328141	315927	302691	8083.44		30	722.84
	PSO	307681	301527	296027	3567.93		100	385.96
	ACO	317040	308187	299019	5227.62		50	601.34
	IACO	302666	299165	295627	2587.69		100	216.30
0.995	GA	320988	284946	256043	23309.32	< 265479	20	772.37
	TSA	292737	274177	255774	11729.56		26	743.66
	PSO	270579	260863	253049	5325.78		76	468.39
	ACO	271993	264604	256134	4609.67		53	533.88
	IACO	266648	259893	252837	4071.65		83	349.02
0.990	GA	241335	218970	192292	15452.78	< 199311	13	843.58
	TSA	228816	213447	191909	11049.87		13	824.05
	PSO	201624	196863	190840	2879.55		73	536.09
	ACO	204391	198872	192056	3471.54		50	563.69
	IACO	199732	194755	189820	3145.28		90	267.66
0.950	GA	173339	157278	138967	11432.89	<143997	13	799.78
	TSA	165863	150538	138982	8074.30		30	730.22
	PSO	145810	141081	137299	2534.56		86	457.34
	ACO	148451	142918	138984	2970.02		67	479.17
	IACO	144479	140406	137140	2633.96		83	319.34
0.900	GA	148754	134971	119257	9811.35	< 123581	13	807.05
	TSA	143239	130157	119507	7363.50		26	738.87
	PSO	124706	121299	118078	2183.52		90	395.56
	ACO	127300	123598	119032	2253.17		43	633.68
	IACO	124030	121141	117696	1995.83		93	291.87
0.850	GA	125952	113350	97787	7926.63	< 101425	10	818.01
	TSA	117671	107426	97656	6972.24		23	768.79
	PSO	102339	99428	97025	1421.84		86	404.14
	ACO	104520	101008	96706	2435.69		53	575.53
	IACO	101409	99296	96595	1491.12		100	212.87

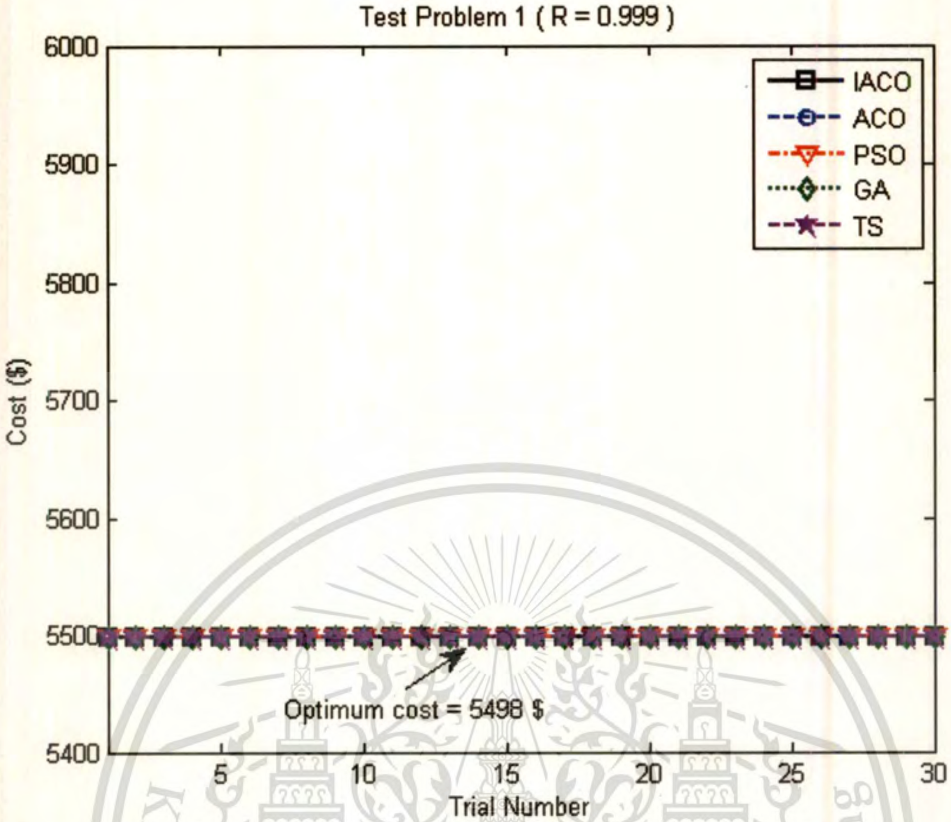


Figure 4.7 Distribution of cost of test problem 1 ($R_0 = 0.999$)

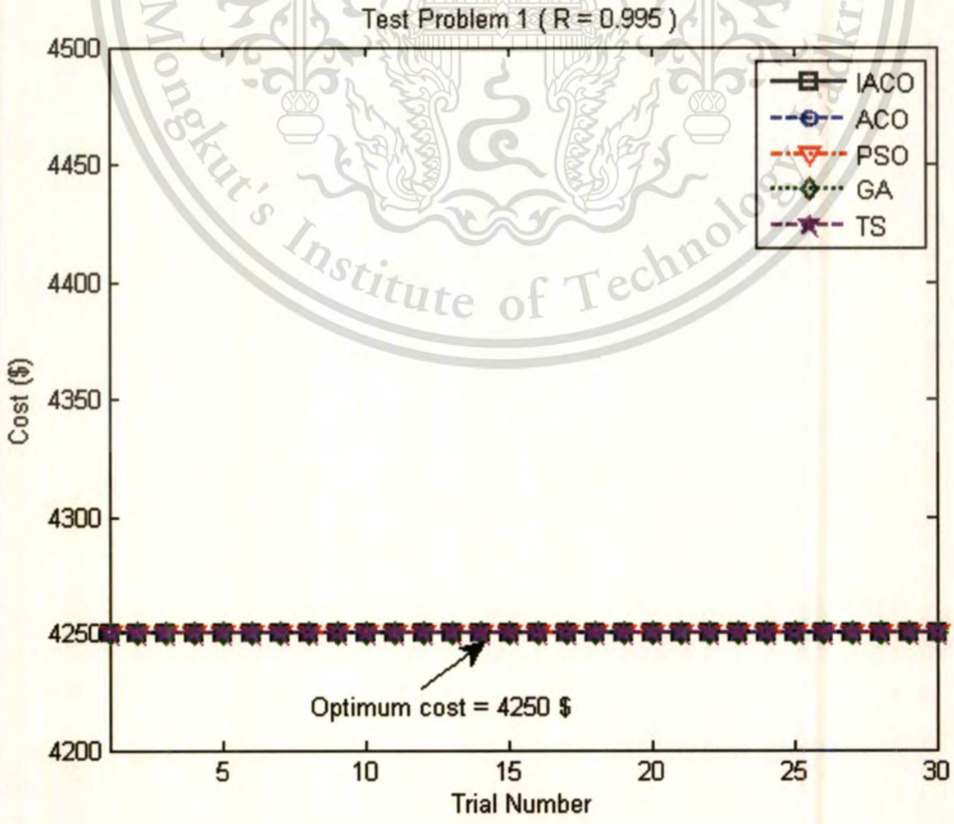


Figure 4.8 Distribution of cost of test problem 1 ($R_0 = 0.995$)

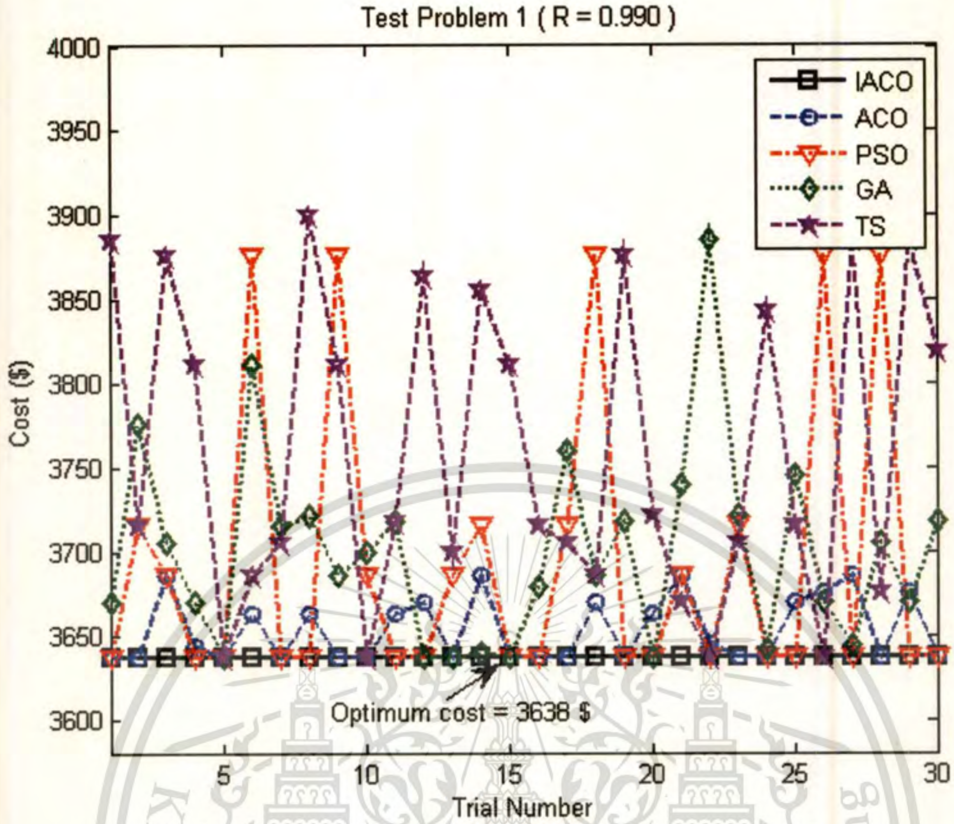


Figure 4.9 Distribution of cost of test problem 1 ($R_0 = 0.990$)

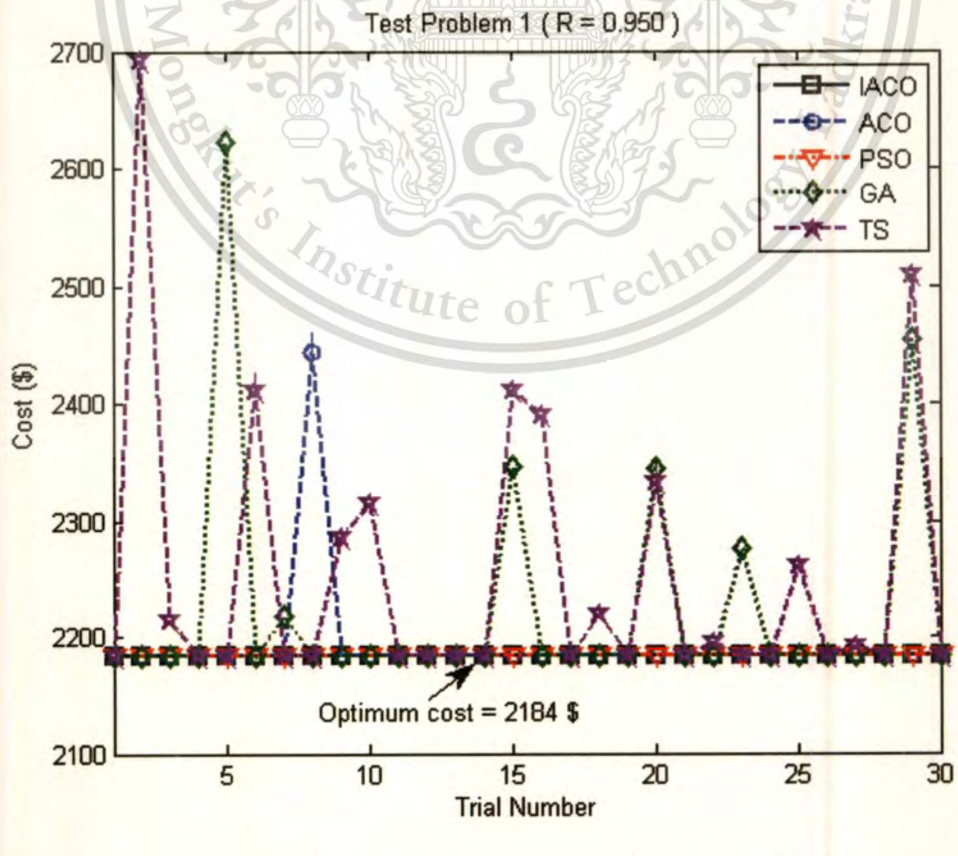


Figure 4.10 Distribution of cost of test problem 1 ($R_0 = 0.950$)

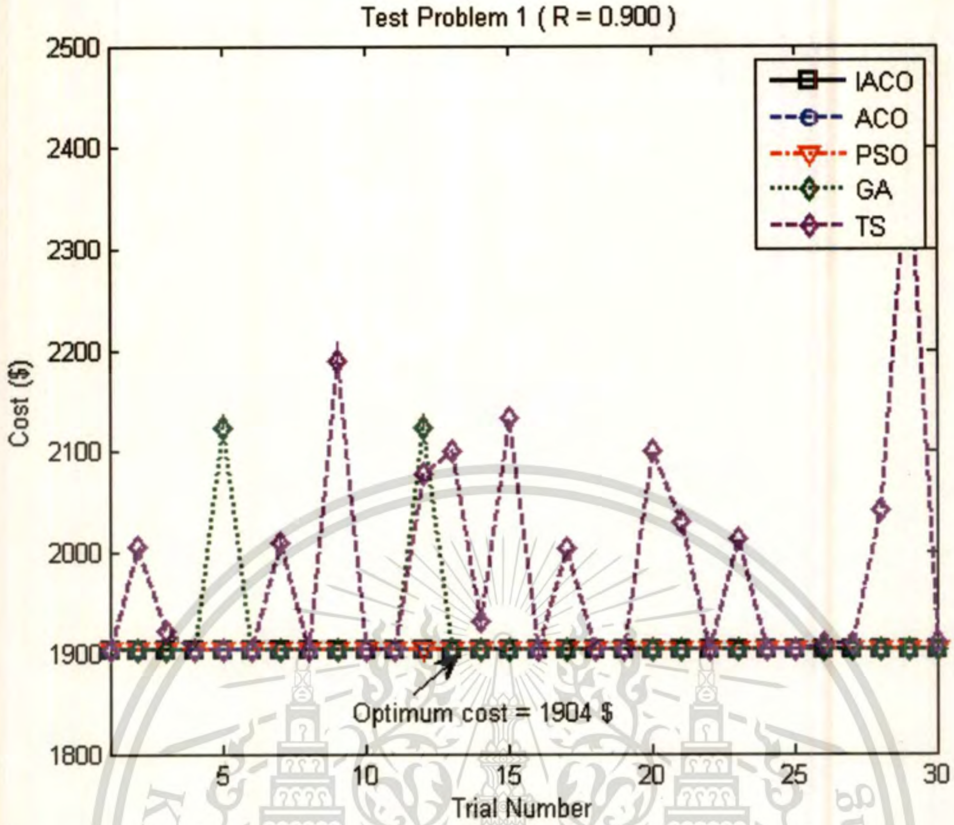


Figure 4.11 Distribution of cost of test problem 1 ($R_0 = 0.900$)

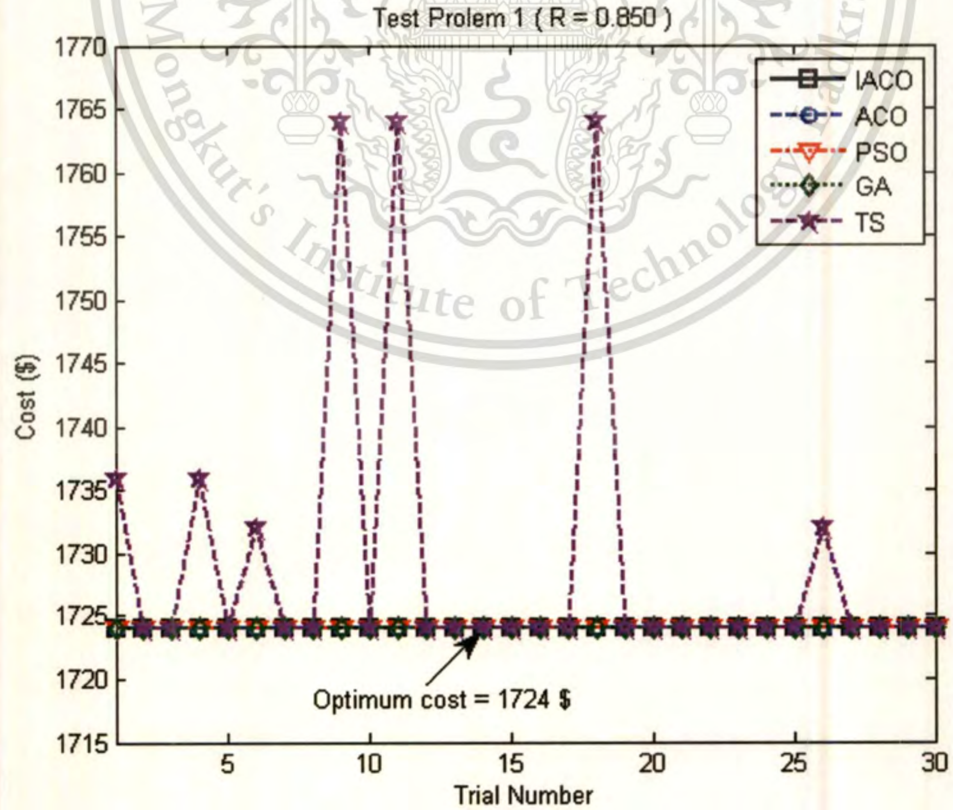


Figure 4.12 Distribution of cost of test problem 1 ($R_0 = 0.850$)

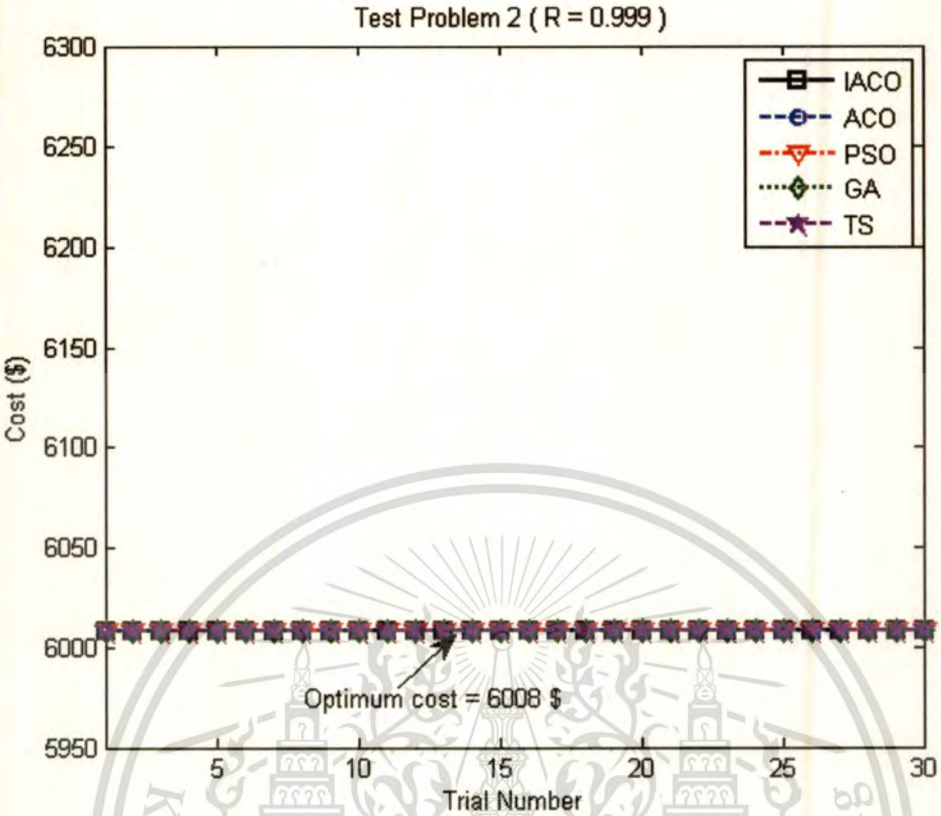


Figure 4.13 Distribution of cost of test problem 2 ($R_0 = 0.999$)

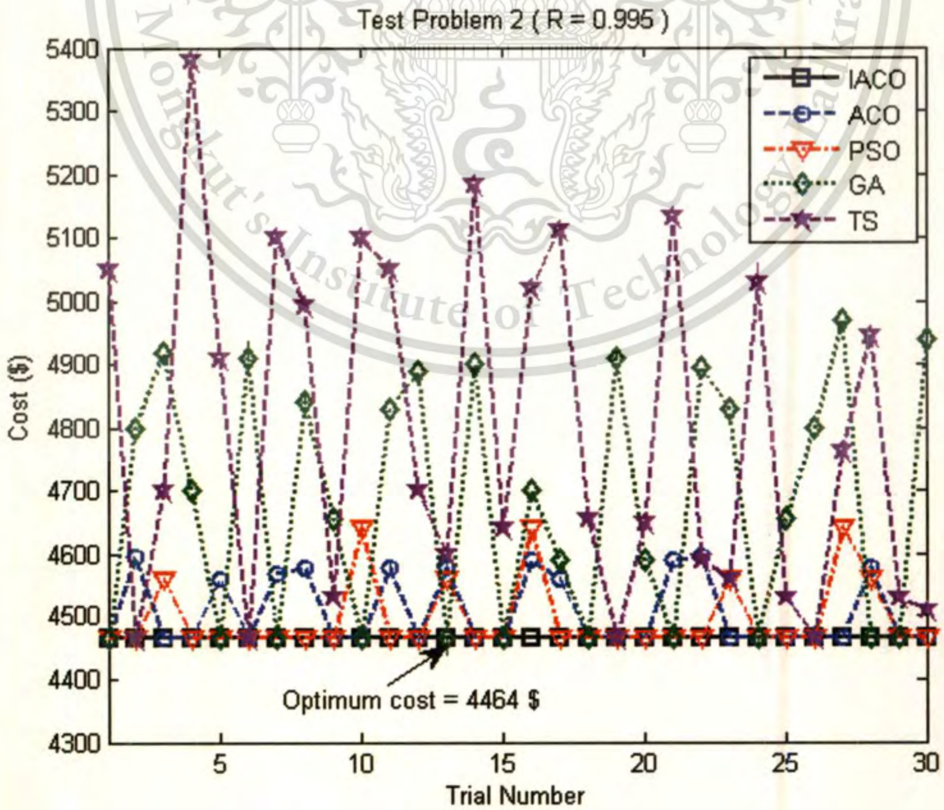


Figure 4.14 Distribution of cost of test problem 2 ($R_0 = 0.995$)

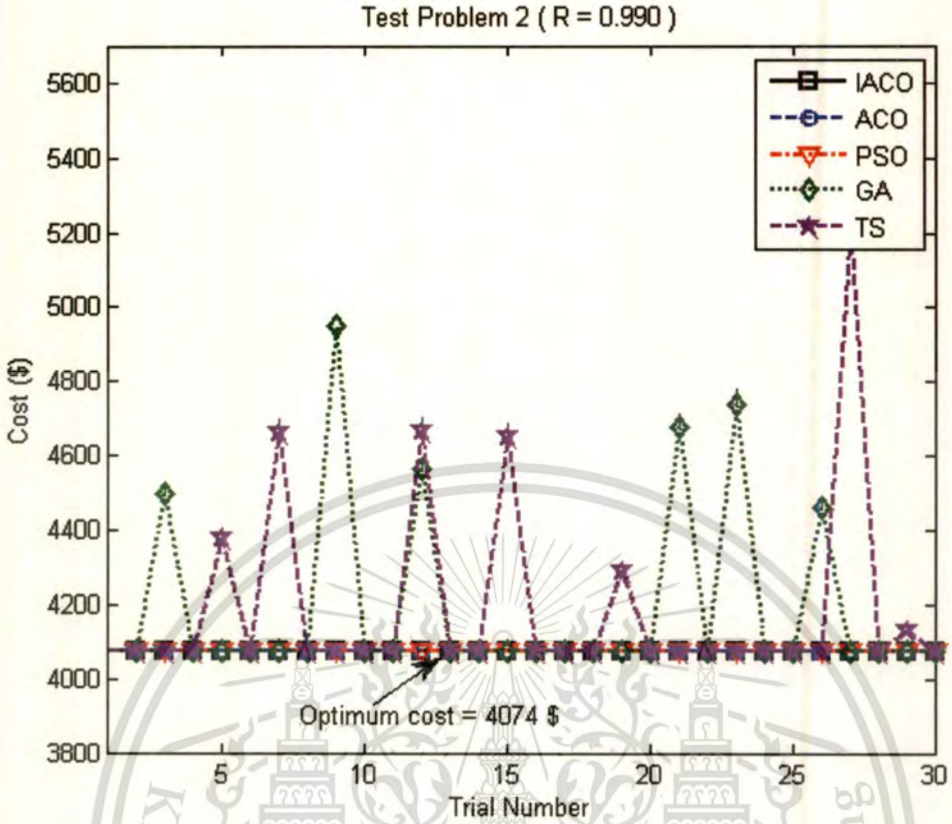


Figure 4.15 Distribution of cost of test problem 2 ($R_0 = 0.990$)

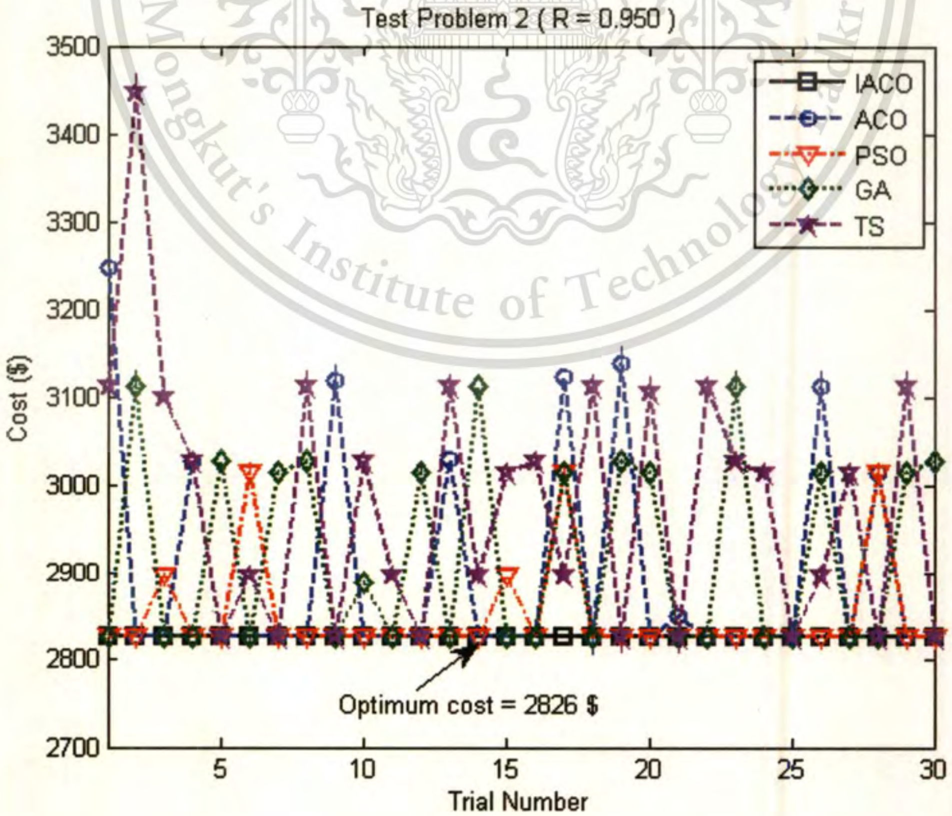


Figure 4.16 Distribution of cost of test problem 2 ($R_0 = 0.950$)

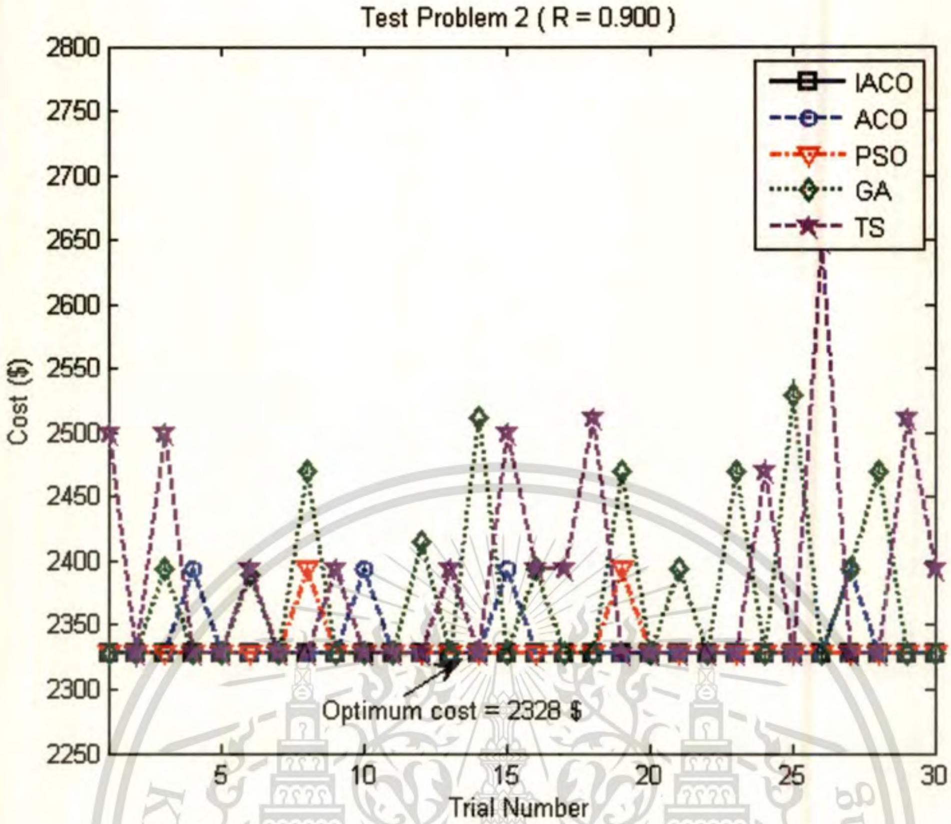


Figure 4.17 Distribution of cost of test problem 2 ($R_0 = 0.900$)

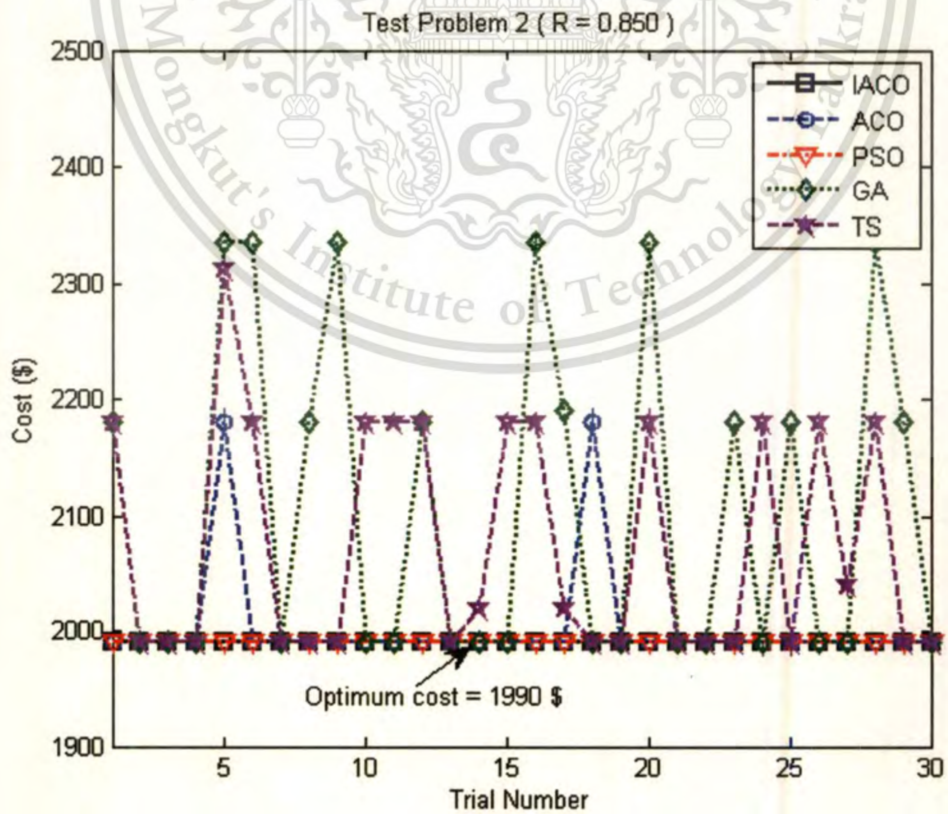


Figure 4.18 Distribution of cost of test problem 2 ($R_0 = 0.850$)

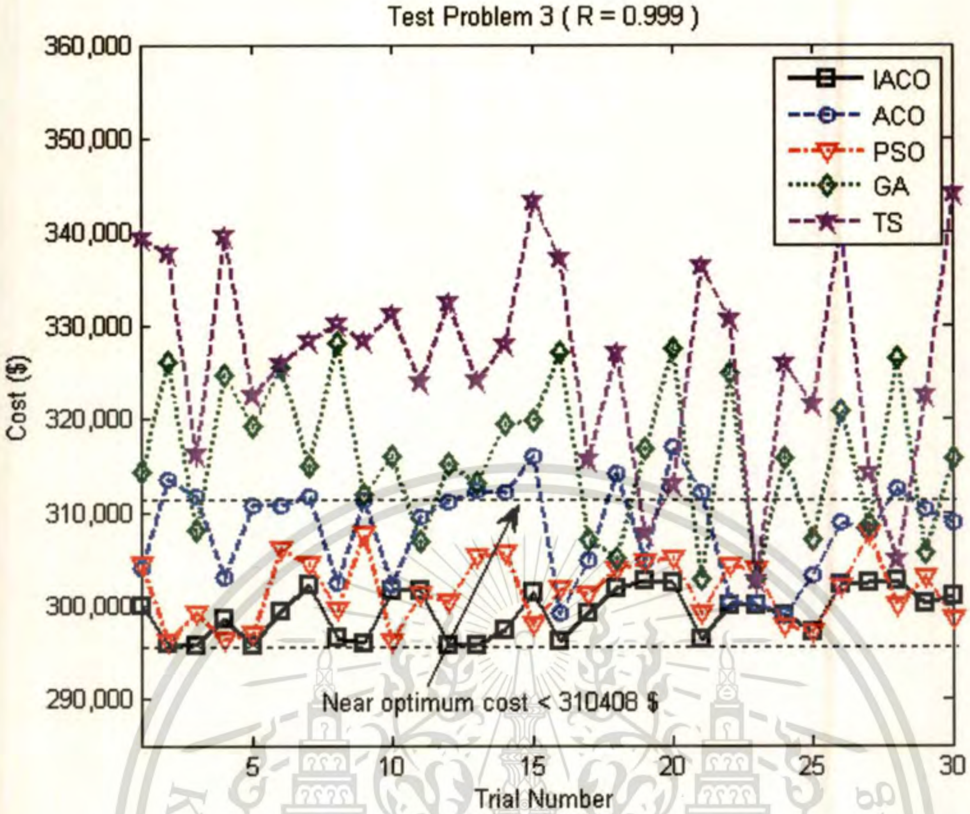


Figure 4.19 Distribution of cost of test problem 3 ($R_0 = 0.999$)

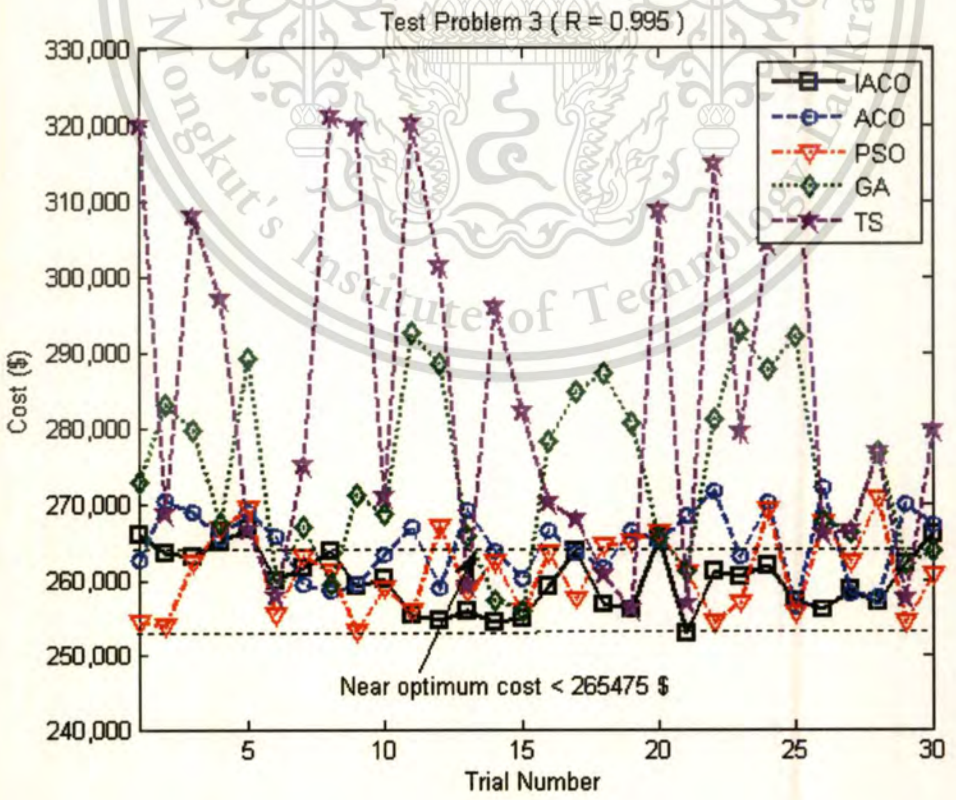


Figure 4.20 Distribution of cost of test problem 3 ($R_0 = 0.995$)

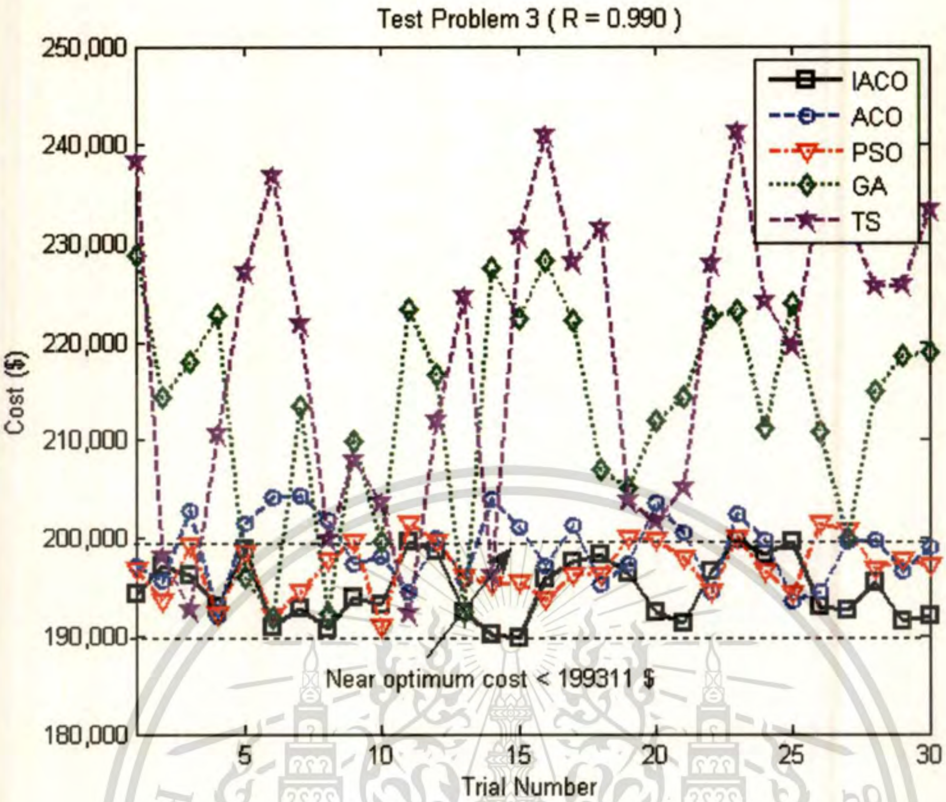


Figure 4.21 Distribution of cost of test problem 3 ($R_0 = 0.990$)

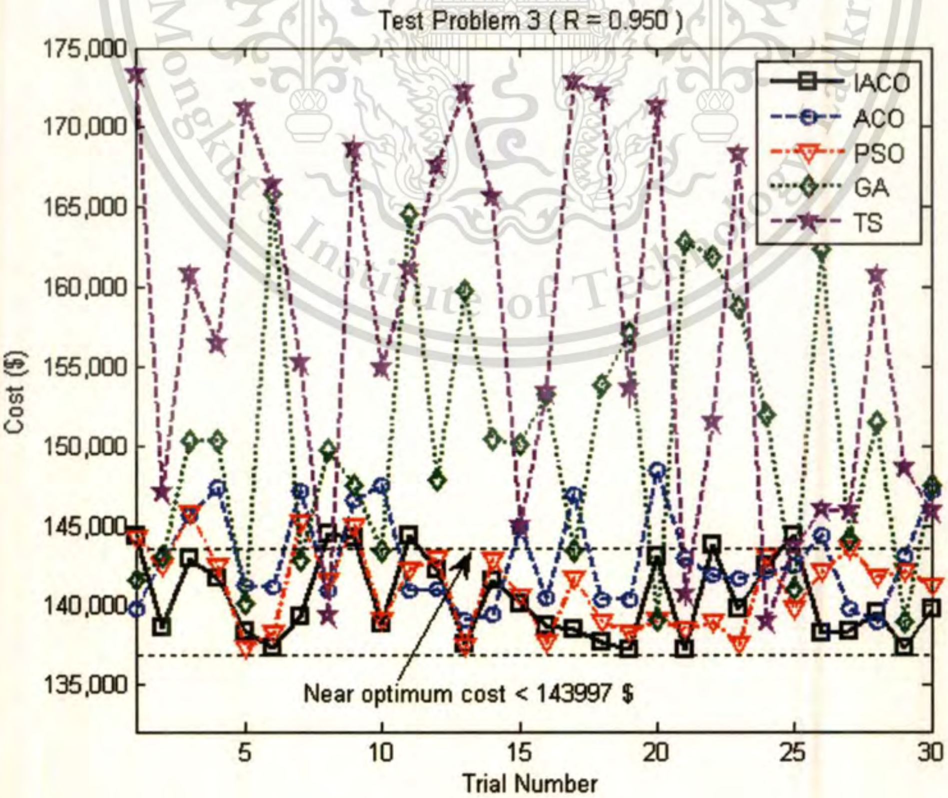


Figure 4.22 Distribution of cost of test problem 3 ($R_0 = 0.950$)

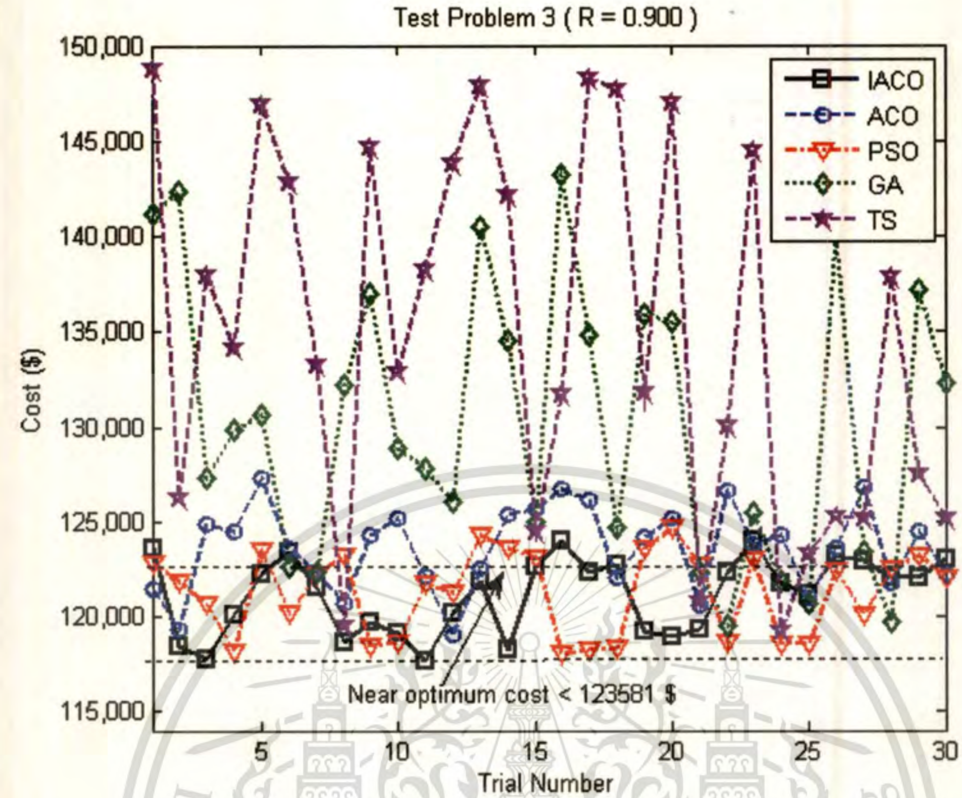


Figure 4.23 Distribution of cost of test problem 3 ($R_0 = 0.900$)

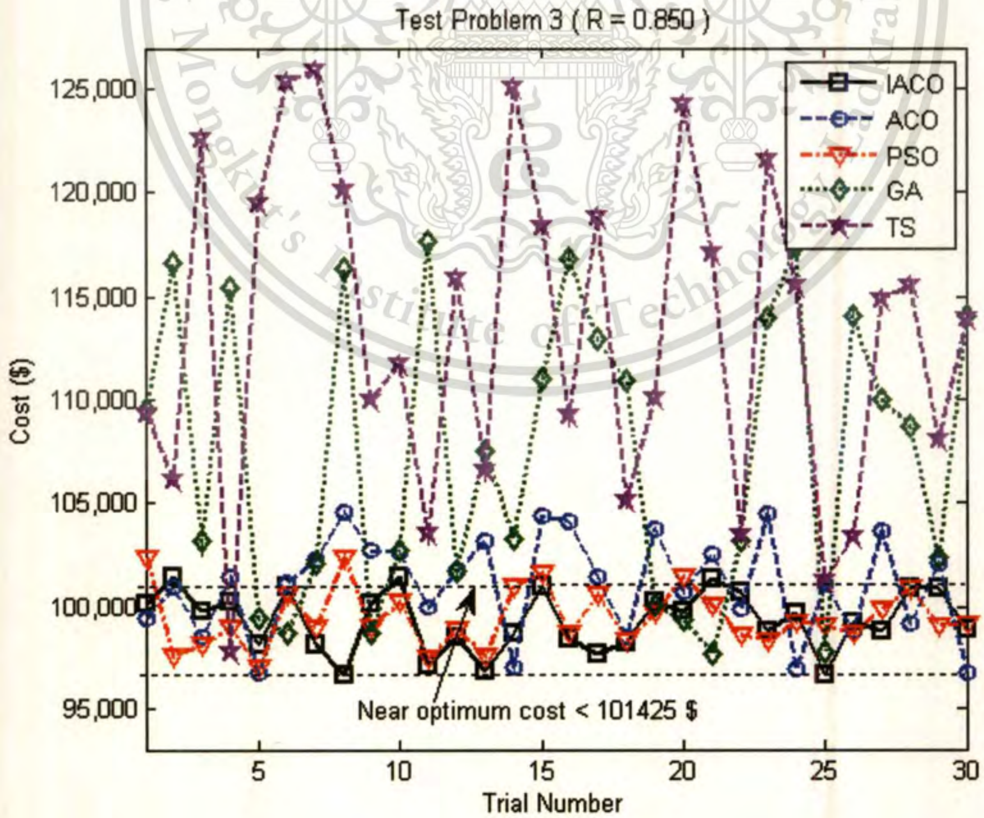


Figure 4.24 Distribution of cost of test problem 3 ($R_0 = 0.850$)

4.4.2 Comparison in Term of Computational Time

To show performance in term of computational time, the examples have been simulated by being extended the time to allow the methods to reach optimum solution in each trial.

Results have been shown in Table 4.7 – 4.9. Figure 4.25 - 4.42 demonstrate computational time comparison to show performance of IACO. Figure 4.45 – 4.60 demonstrate the distribution outlines of the best solution of each trial for 6 reliability levels in test problem 1, 2 and 3, respectively. Although all methods could find better solution when extended time has been given, IACO still get the better solutions than those methods. Besides, when network is larger, GA, TSA, PSO, ACO could not always guarantee the optimum solutions in some reliability constraints but IACO.

Table 4.7 Summary results of a five-node network problem (performed 120 seconds)

Reliability	Optimal cost ($)	Configuration	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	% Get optimal	CPU Time (s)
0.999	**5498**	**3323323333**	**GA**	5498	5498	5498	0.00000	100	13.13
			TSA	5498	5498	5498	0.00000	100	9.95
			PSO	5530	5498	5498	0.00000	100	5.64
			ACO	5530	5498	5498	0.00000	100	7.79
			IACO	5498	5498	5498	0.00000	100	4.80
0.995	**4250**	**3303222313**	**GA**	4250	4250	4250	0.00000	100	42.63
			TSA	4250	4250	4250	0.00000	100	40.20
			PSO	4250	4250	4250	0.00000	100	25.27
			ACO	4250	4250	4250	0.00000	100	32.36
			IACO	4250	4250	4250	0.00000	100	18.95
0.990	**3638**	**3203312303**	**GA**	3638	3638	3638	0.00000	100	70.03
			TSA	3638	3638	3638	0.00000	100	80.88
			PSO	3638	3638	3638	0.00000	100	30.26
			ACO	3638	3638	3638	0.00000	100	42.06
			IACO	3638	3638	3638	0.00000	100	22.58
0.950	**2184**	**3003300303**	**GA**	2184	2184	2184	0.00000	100	42.38
			TSA	2184	2184	2184	0.00000	100	32.03
			PSO	2184	2184	2184	0.00000	100	15.36
			ACO	2184	2184	2184	0.00000	100	20.24
			IACO	2184	2184	2184	0.00000	100	11.28
0.900	**1904**	**3003200203**	**GA**	1904	1904	1904	0.00000	100	41.20
			TSA	1904	1904	1904	0.00000	100	20.52
			PSO	1904	1904	1904	0.00000	100	12.03
			ACO	1904	1904	1904	0.00000	100	15.21
			IACO	1904	1904	1904	0.00000	100	5.79
0.850	**1724**	**2003300102**	**GA**	1724	1724	1724	0.00000	100	43.57
			TSA	1724	1724	1724	0.00000	100	15.63
			PSO	1724	1724	1724	0.00000	100	11.85
			ACO	1724	1724	1724	0.00000	100	8.11
			IACO	1724	1724	1724	0.00000	100	2.09

Table 4.8 Summary results of a source-destination network problem (performed 240 seconds)

Reliability	Optimal cost ($)	Configuration	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	% Get optimal	CPU Time (s)
0.999	6008	033300000001013330	GA	6808	6008	6008	0.000000	100	73.61
			TSA	6008	6008	6008	0.000000	100	59.65
			PSO	6008	6008	6008	0.000000	100	25.91
			ACO	6008	6008	6008	0.000000	100	34.49
			IACO	6008	6008	6008	0.000000	100	21.78
0.995	4464	022200000000003330	GA	5112	4795	4464	182.297168	46	187.58
			TSA	4910	4572	4464	168.931204	66	143.02
			PSO	4640	4476	4464	39.641149	90	54.07
			ACO	4596	4487	4464	47.606094	80	100.70
			IACO	4464	4464	4464	0.000000	100	18.03
0.990	4074	013100000000001330	GA	4670	4148	4074	184.235294	83	104.37
			TSA	4565	4117	4074	133.256218	90	88.26
			PSO	4074	4074	4074	0.000000	100	35.77
			ACO	4074	4074	4074	0.000000	100	42.68
			IACO	4074	4074	4074	0.000000	100	26.47
0.950	2826	002100000000000330	GA	3106	2902	2826	97.521563	53	170.32
			TSA	3028	2874	2826	83.613369	73	152.36
			PSO	2898	2831	2826	18.266985	93	58.89
			ACO	3120	2856	2826	78.275699	83	125.41
			IACO	2826	2826	2826	0.000000	100	40.60
0.900	2328	011000000000001300	GA	2648	2357	2328	65.432566	73	160.45
			TSA	2470	2351	2328	46.132294	76	113.51
			PSO	2328	2328	2328	0.000000	100	42.27
			ACO	2470	2335	2328	27.928974	93	72.97
			IACO	2328	2328	2328	0.000000	100	30.39
0.850	1990	002000001000000220	GA	2180	2019	1990	64.910194	73	106.70
			TSA	2180	2011	1990	57.795716	83	102.49
			PSO	1990	1990	1990	0.000000	100	53.43
			ACO	1990	1990	1990	0.000000	100	77.79
			IACO	1990	1990	1990	0.000000	100	46.63

Table 4.9 Summary results of a19 districts in Bangkok, Thailand (performed 1800 seconds)

Reliability	Algorithm	Max Cost ($)	Average Cost ($)	Min Cost ($)	Standard Deviation	Near Optimal cost ($)	% Get near optimal	CPU Time (s)
0.999	GA	329607	316844	302007	8609.87	< 310408	26	1499.83
	TSA	327760	313629	301285	7861.02		36	1419.75
	PSO	302917	300208	298437	1362.64		100	346.20
	ACO	314264	305955	297459	5544.63		73	891.40
	IACO	302666	299165	295627	2587.69		100	226.59
0.995	GA	315879	283272	255128	17209.67	< 265479	20	1615.24
	TSA	289021	271160	255311	10636.54		23	1540.61
	PSO	265427	259379	253318	3733.48		100	476.02
	ACO	269946	262383	256299	4146.16		73	896.32
	IACO	262359	258957	253187	3082.89		100	498.04
0.990	GA	214550	202209	190856	6994.77	< 199311	36	1393.16
	TSA	215384	202684	191721	6703.28		36	1412.28
	PSO	201300	196230	191149	3129.56		80	708.36
	ACO	200997	195499	191104	3151.52		76	886.64
	IACO	198373	194006	190104	2531.64		100	261.03
0.950	GA	150191	144137	139019	3604.82	<143997	43	1253.32
	TSA	152505	145043	139144	3886.68		40	1314.86
	PSO	143040	141024	137718	1551.85		100	421.66
	ACO	145386	142253	139340	1926.92		73	842.80
	IACO	143425	140798	137933	1572.71		100	369.12
0.900	GA	133455	127957	120125	4207.30	< 123581	23	1537.88
	TSA	131233	125467	119784	3428.98		30	1415.56
	PSO	124154	121279	118092	1914.39		86	583.12
	ACO	126260	122960	118356	2448.21		43	1202.14
	IACO	121863	120014	117728	1341.77		100	274.42
0.850	GA	121384	108135	98306	7697.44	< 101425	33	1479.22
	TSA	109554	104689	99263	3537.11		30	1519.75
	PSO	101518	99688	97162	1358.36		90	569.03
	ACO	103237	100082	96823	1947.06		66	1080.72
	IACO	101369	99135	96597	1416.23		100	265.54

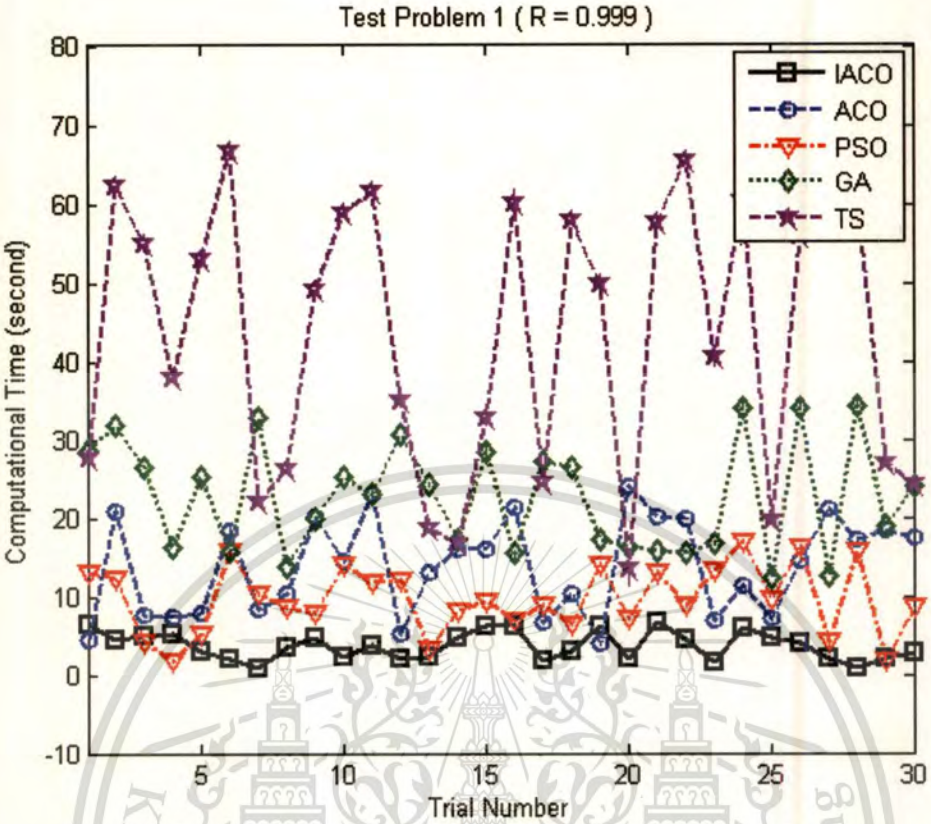


Figure 4.25 Computational time of test problem 1 ($R_0 = 0.999$)

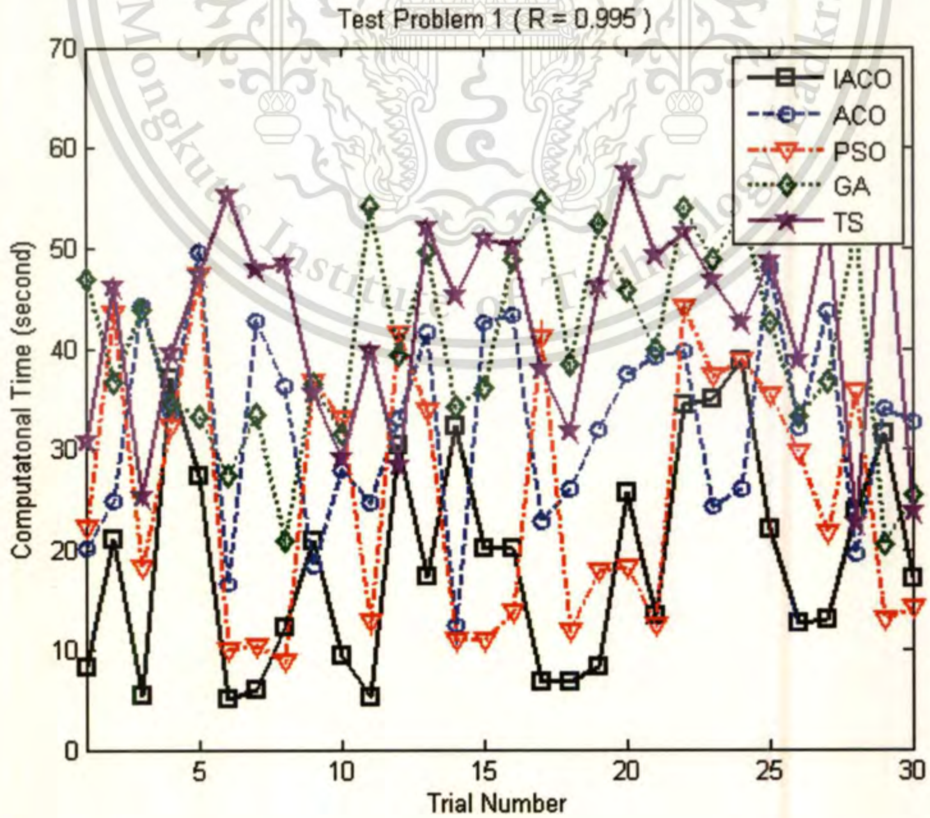


Figure 4.26 Computational time of test problem 1 ($R_0 = 0.995$)

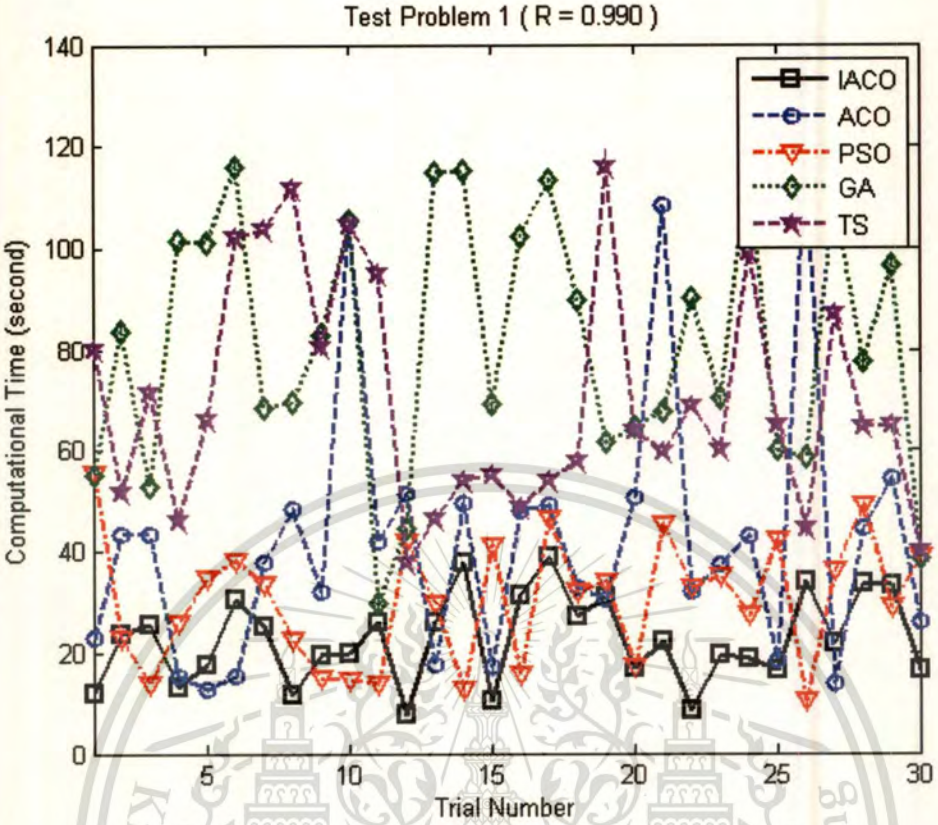


Figure 4.27 Computational time of test problem 1 ($R_0 = 0.990$)

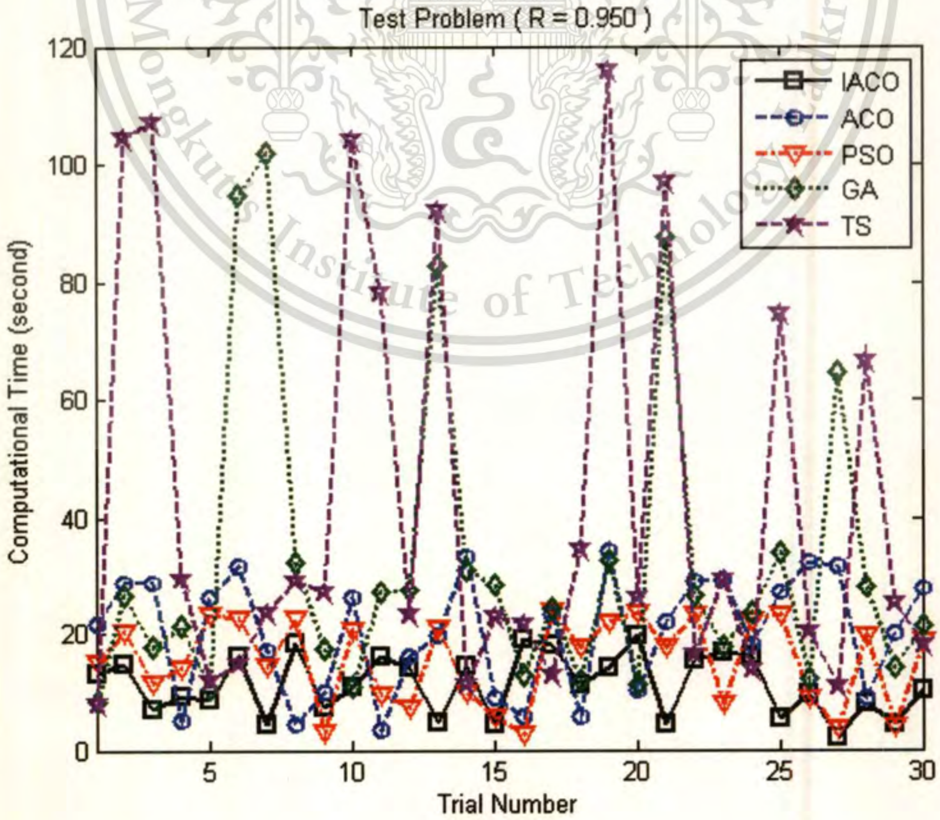


Figure 4.28 Computational time of test problem 1 ($R_0 = 0.950$)

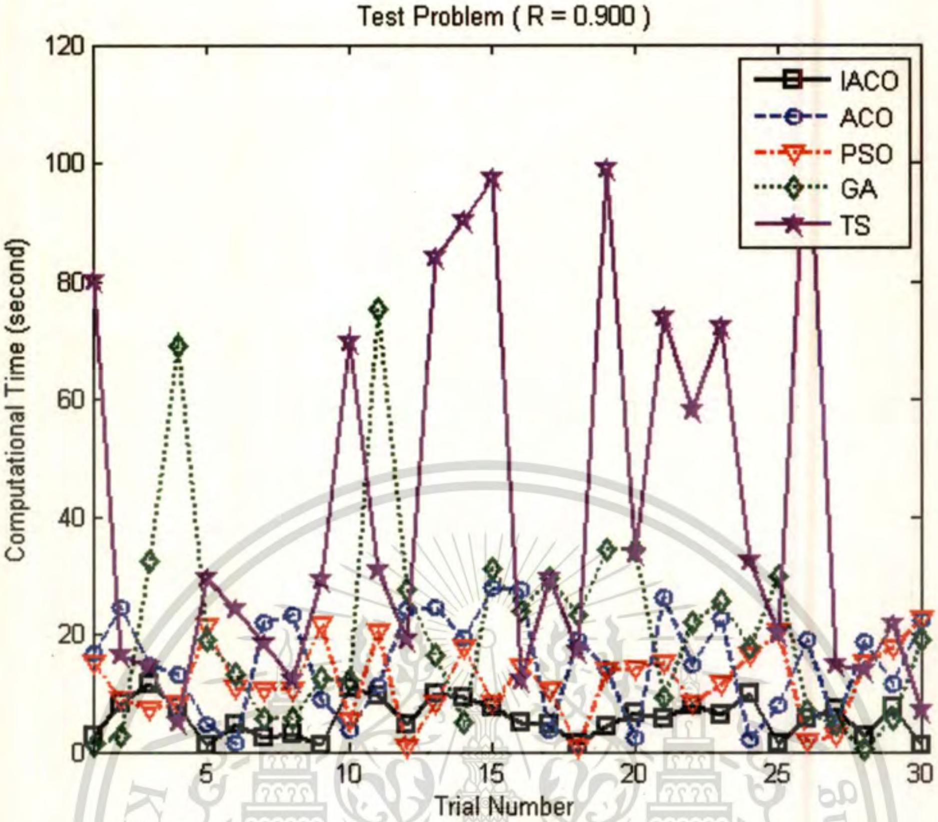


Figure 4.29 Computational time of test problem 1 ($R_0 = 0.900$)

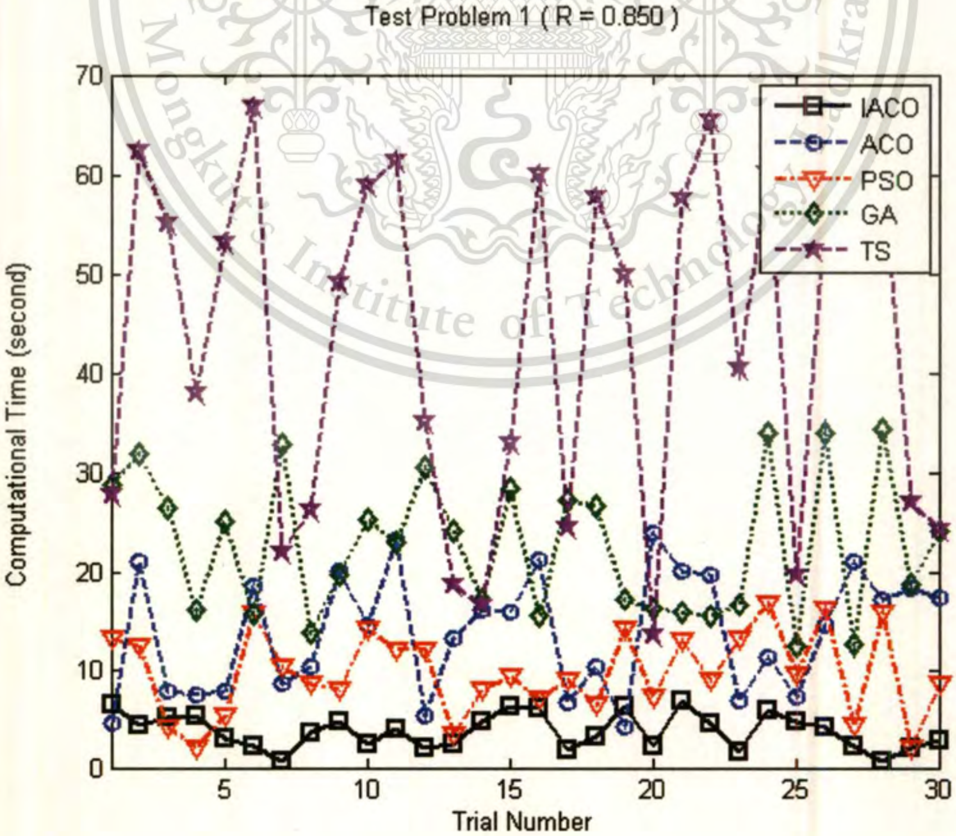


Figure 4.30 Computational time of test problem 1 ($R_0 = 0.850$)

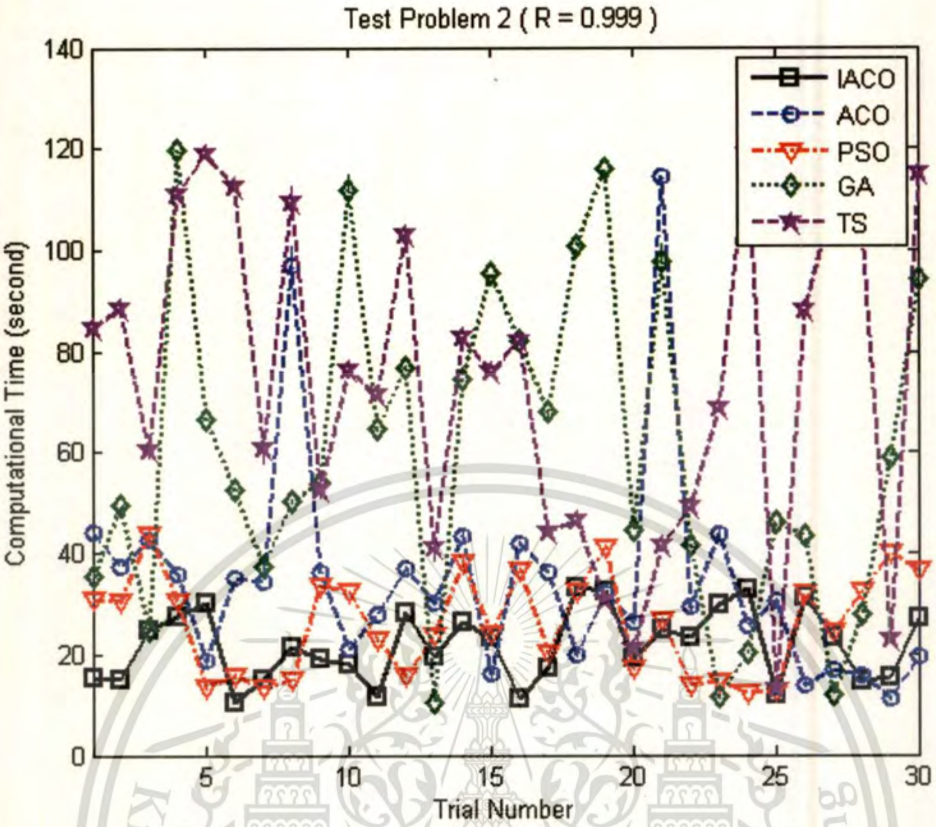


Figure 4.31 Computational time of test problem 2 ($R_0 = 0.999$)

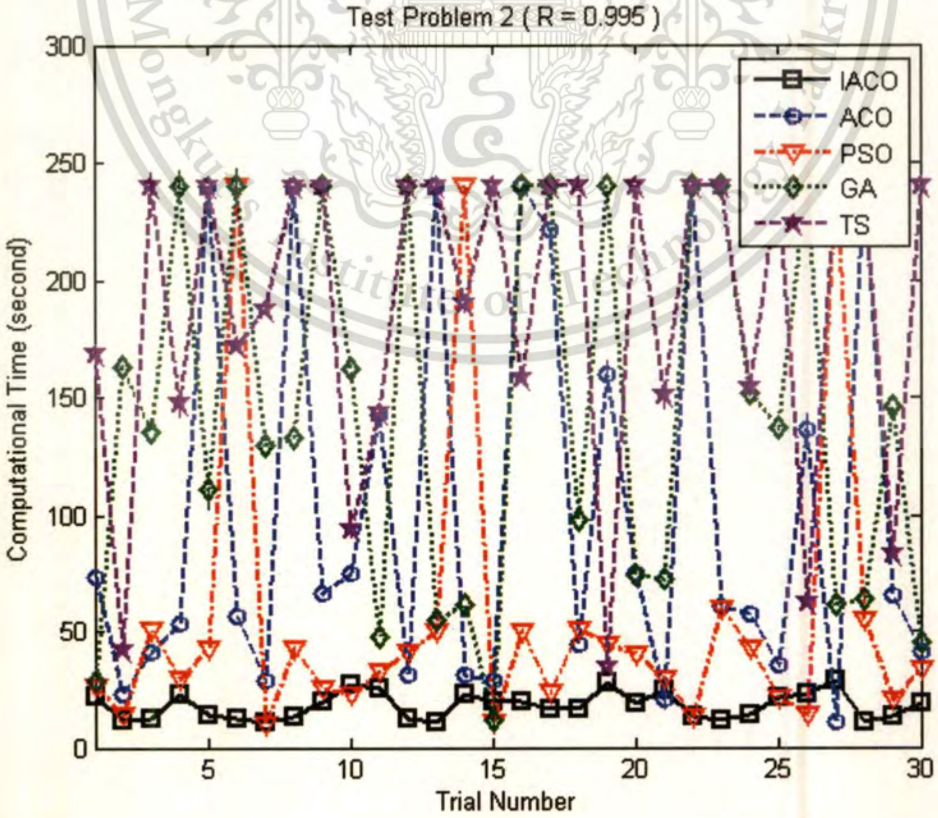


Figure 4.32 Computational time of test problem 2 ($R_0 = 0.995$)

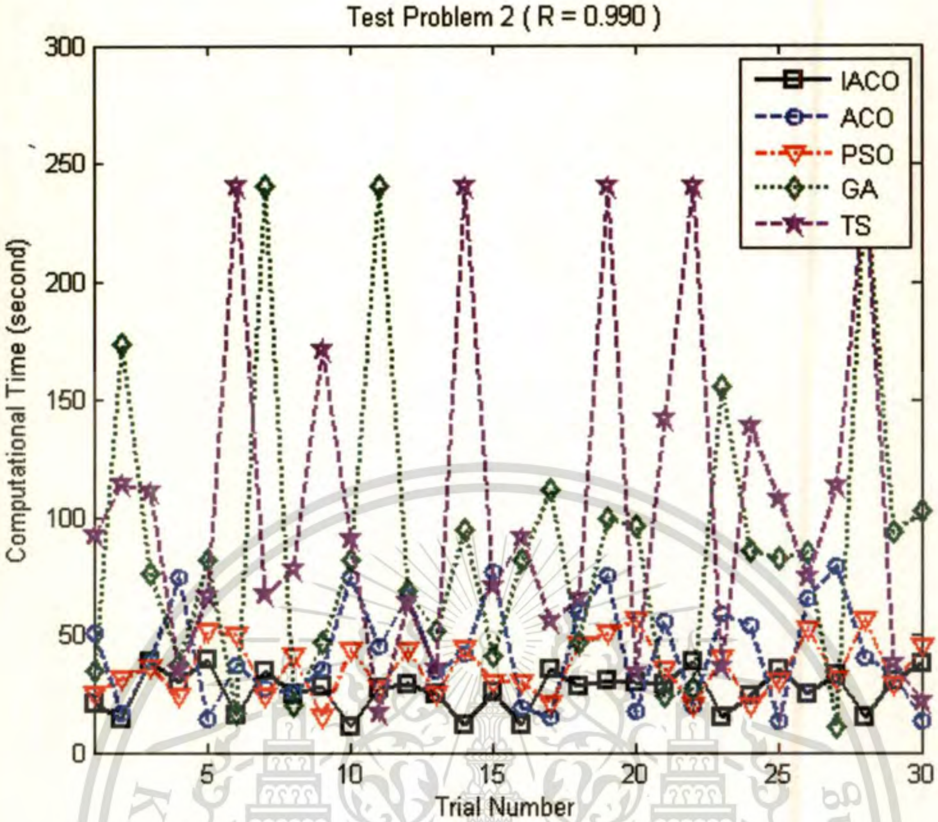


Figure 4.33 Computational time of test problem 2 ($R_0 = 0.990$)

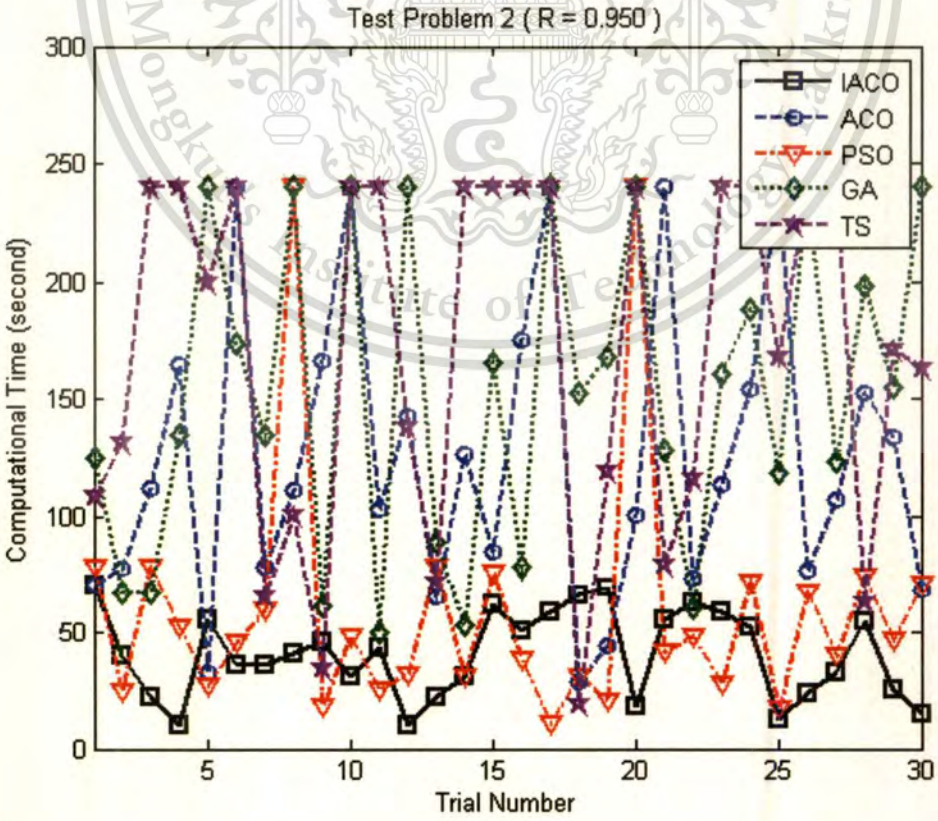


Figure 4.34 Computational time of test problem 2 ($R_0 = 0.950$)

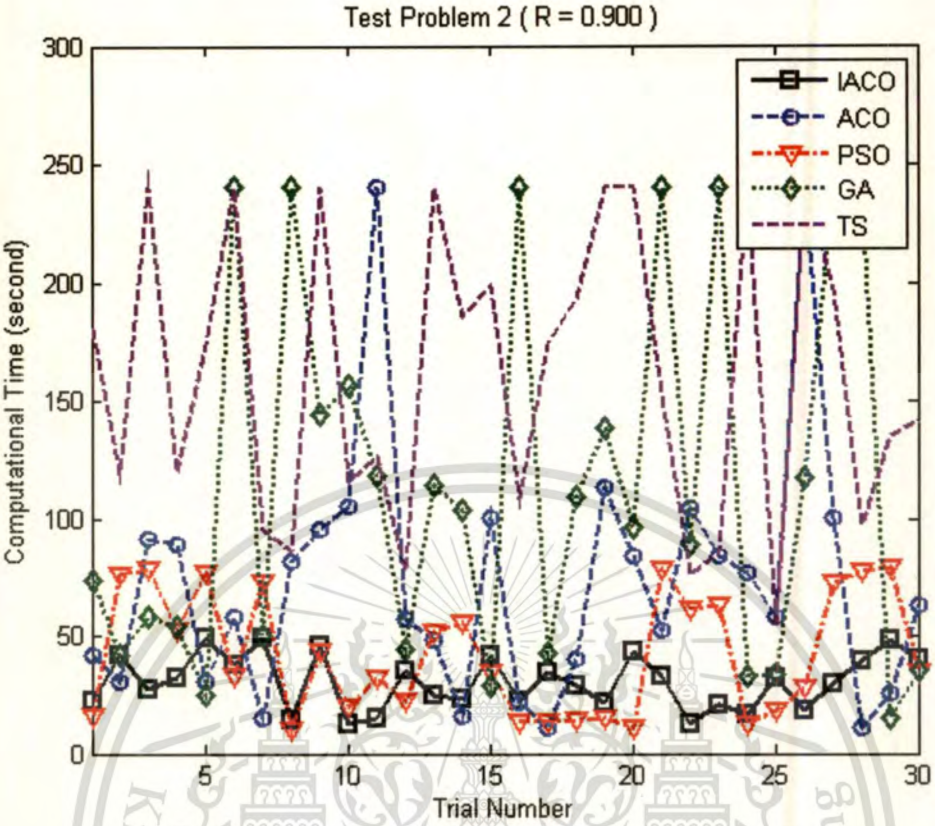


Figure 4.35 Computational time of test problem 2 ($R_0 = 0.900$)

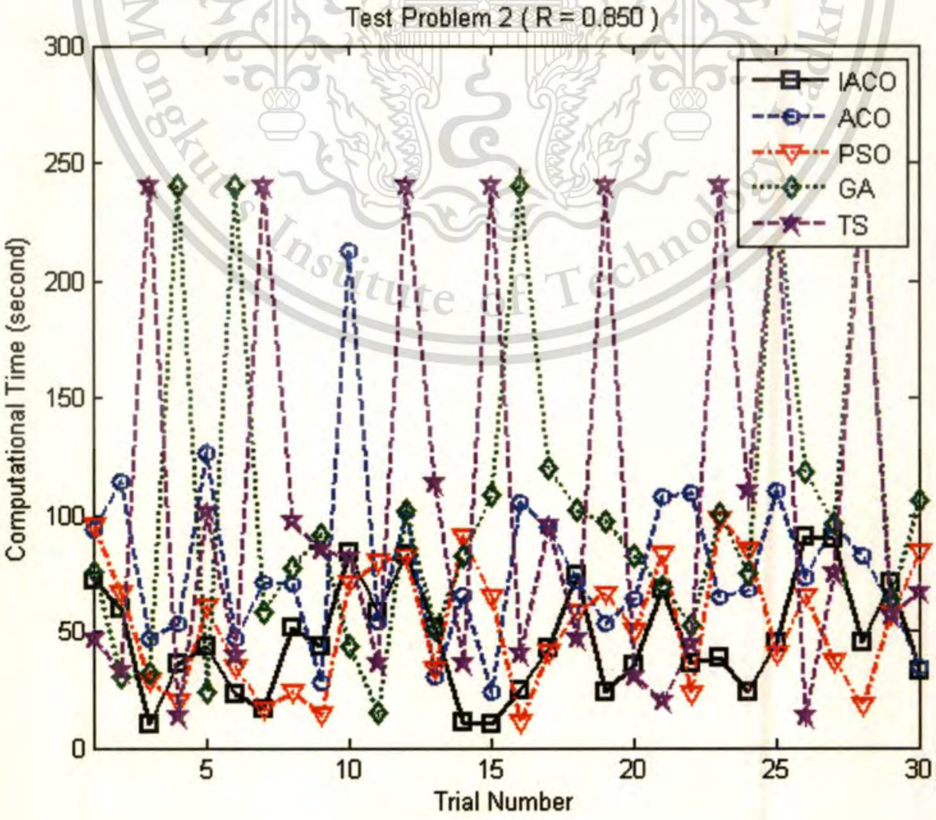


Figure 4.36 Computational time of test problem 2 ($R_0 = 0.850$)

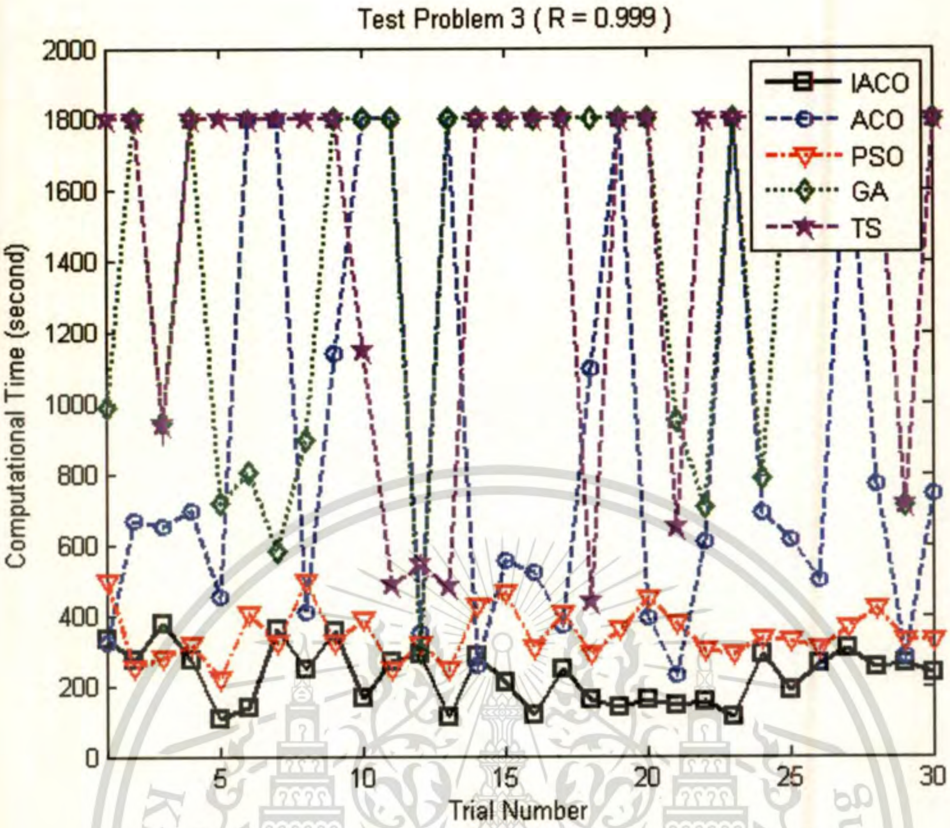


Figure 4.37 Computational time of test problem 3 ($R_0 = 0.999$)

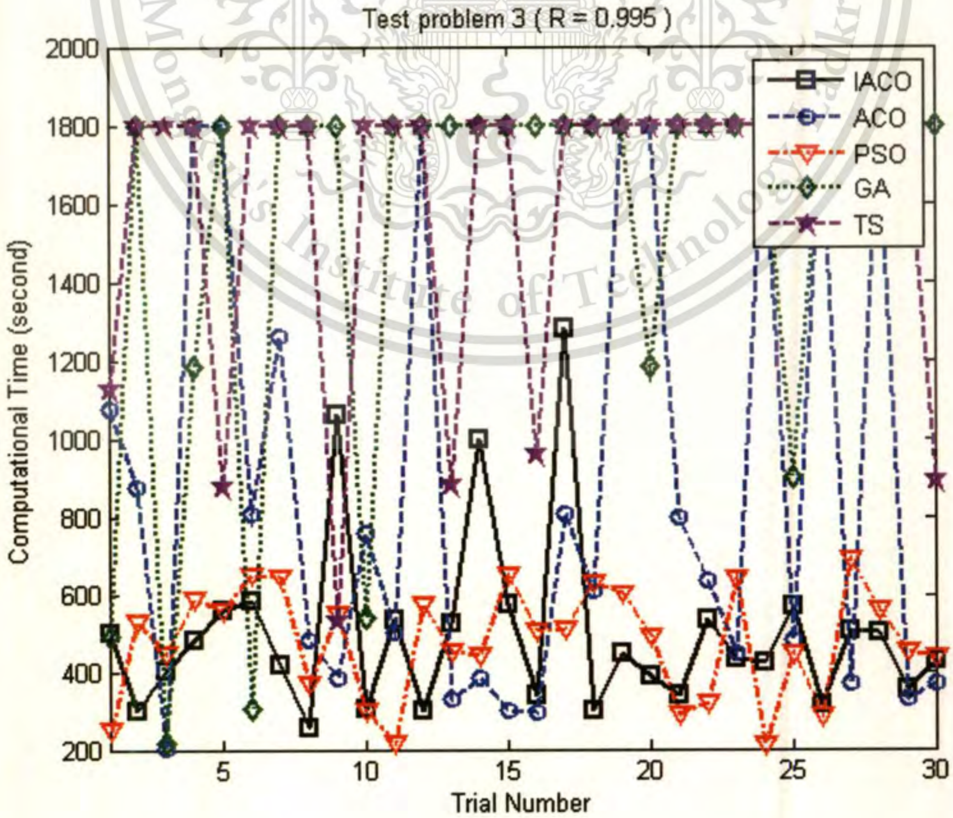


Figure 4.38 Computational time of test problem 3 ($R_0 = 0.995$)

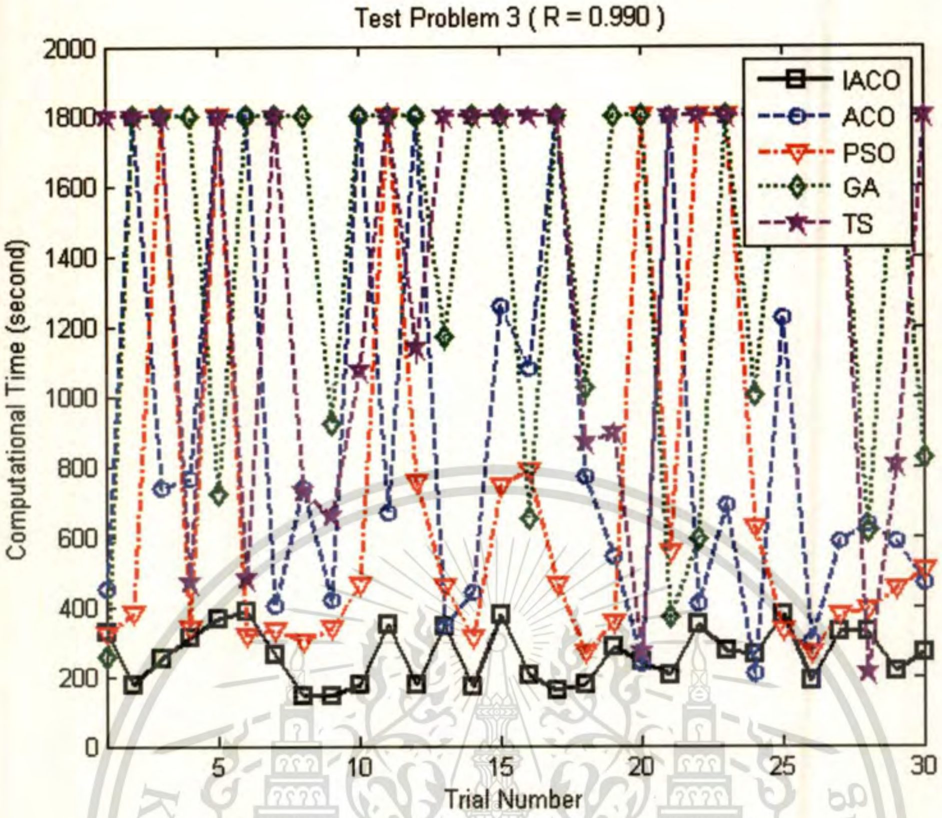


Figure 4.39 Computational time of test problem 3 ($R_0 = 0.990$)

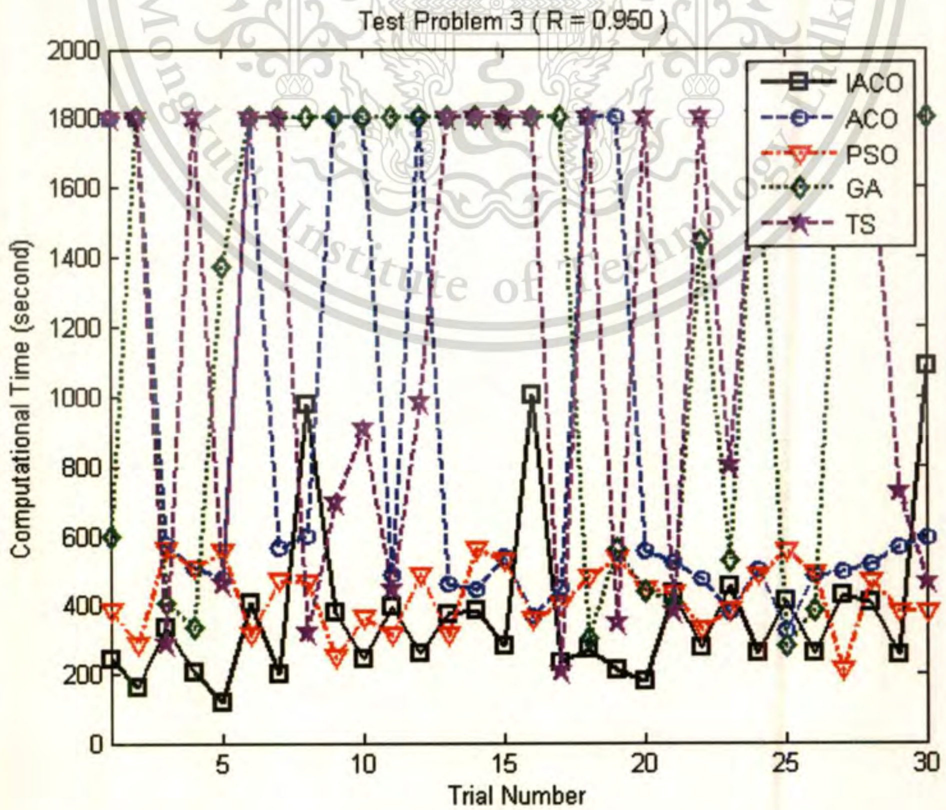


Figure 4.40 Computational time of test problem 3 ($R_0 = 0.950$)

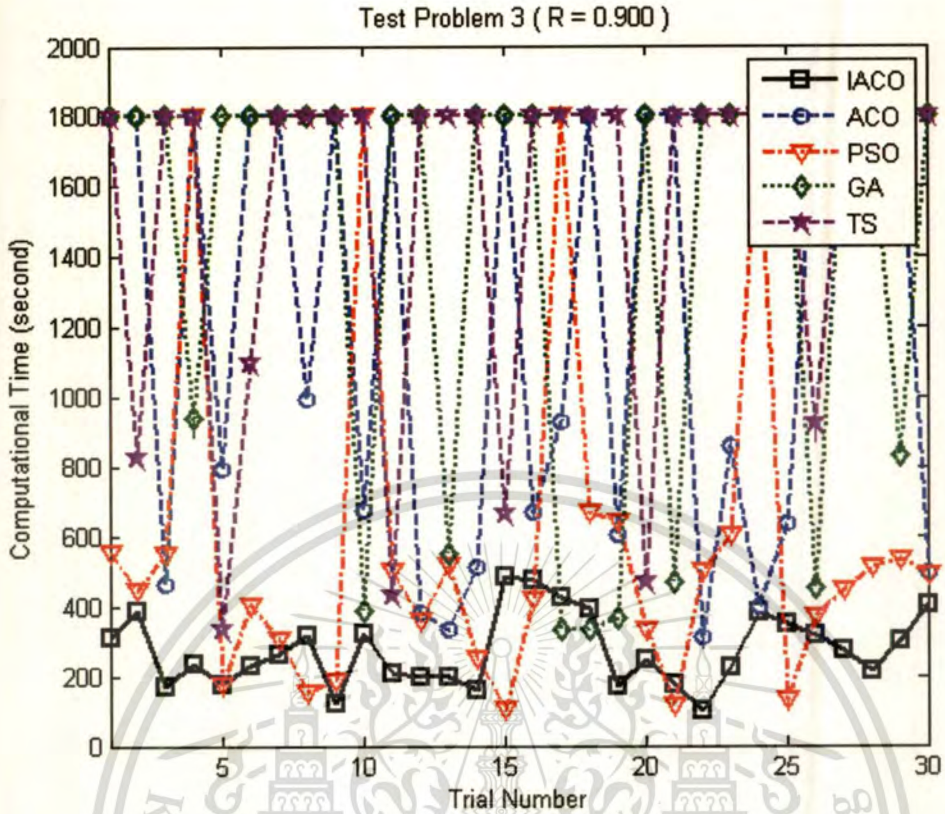


Figure 4.41 Computational time of test problem 3 (R_0 = 0.900)

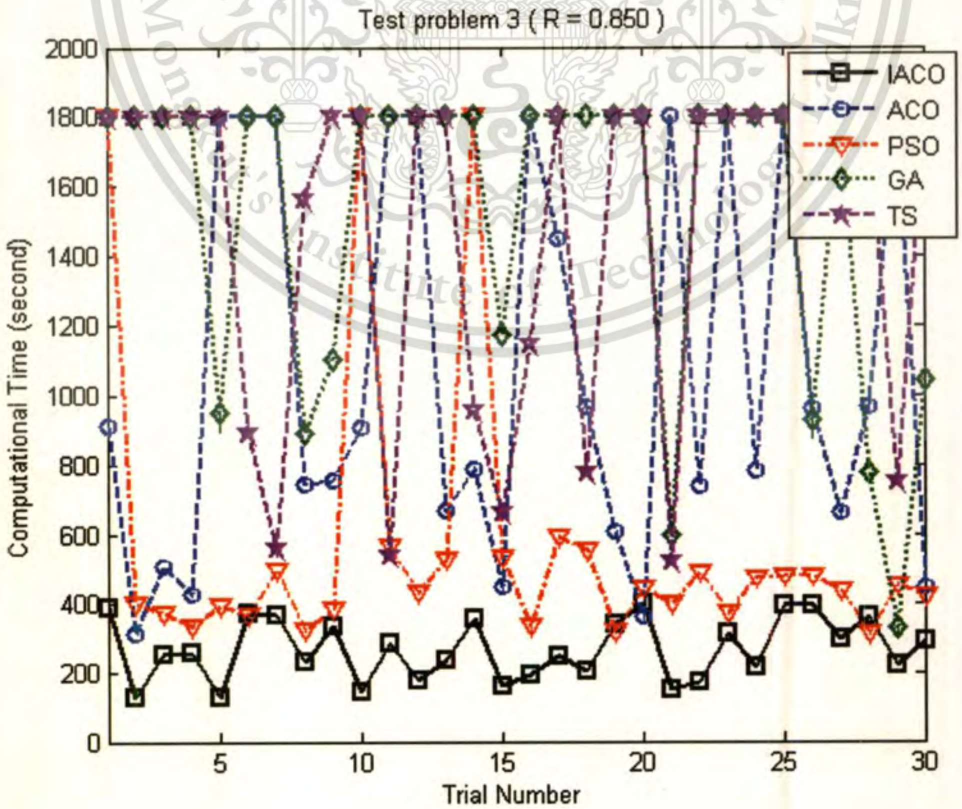


Figure 4.42 Computational time of test problem 3 (R_0 = 0.850)

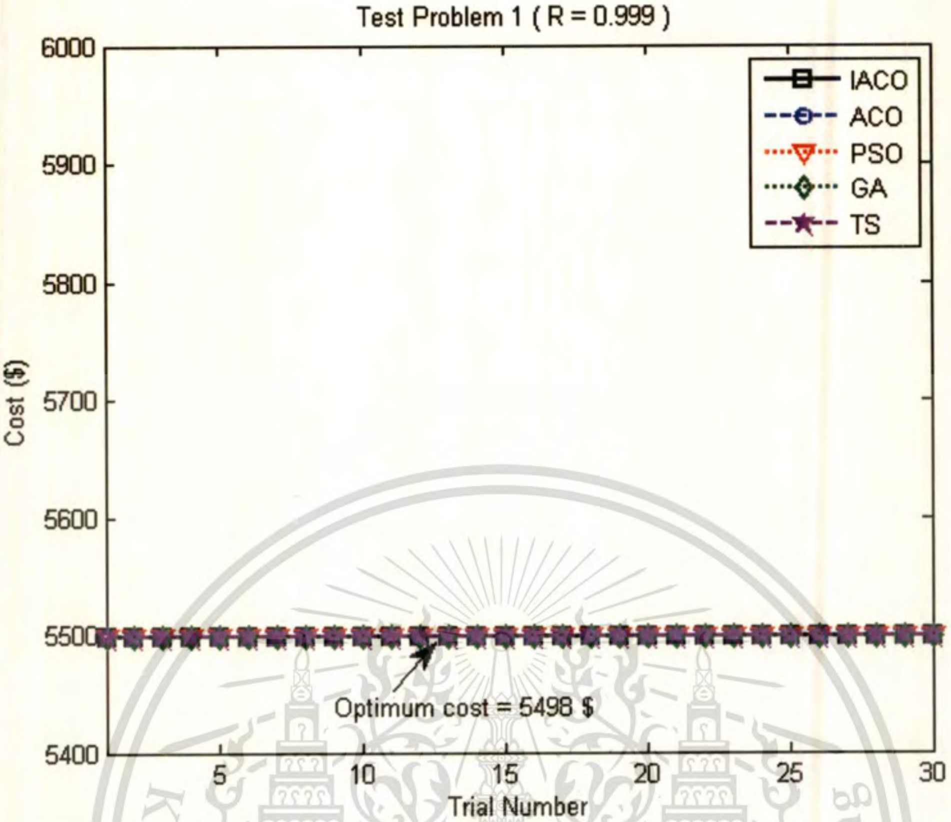


Figure 4.43 Distribution of cost of test problem 1 ($R_0 = 0.999$)

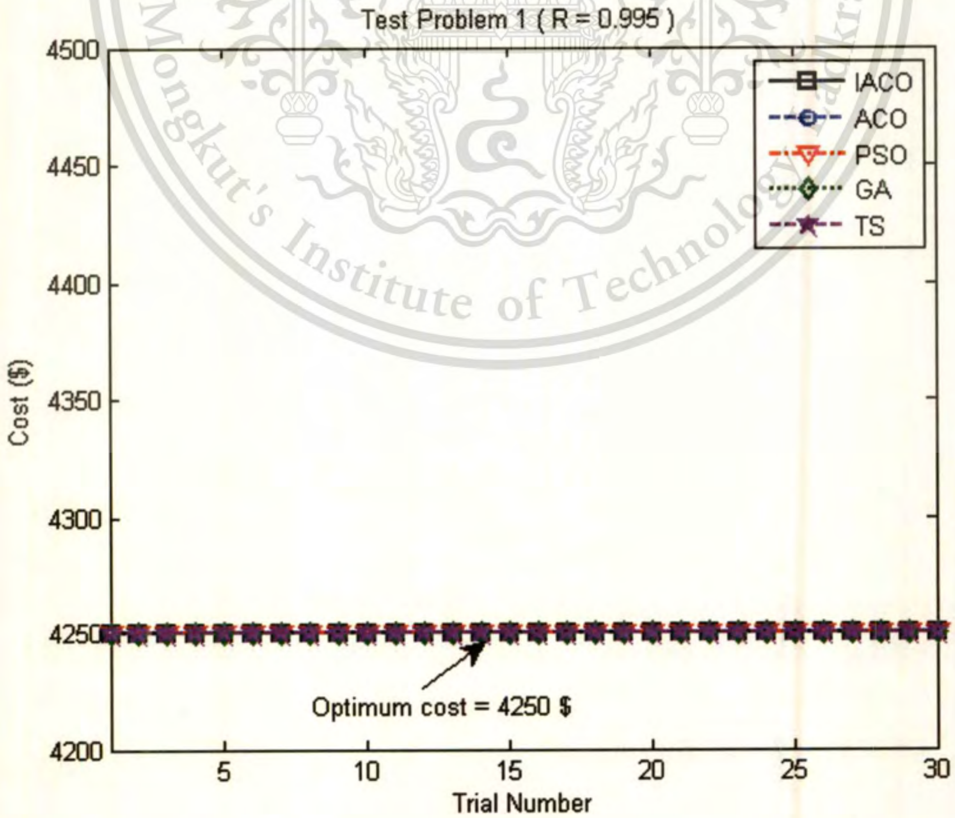


Figure 4.44 Distribution of cost of test problem 1 ($R_0 = 0.995$)

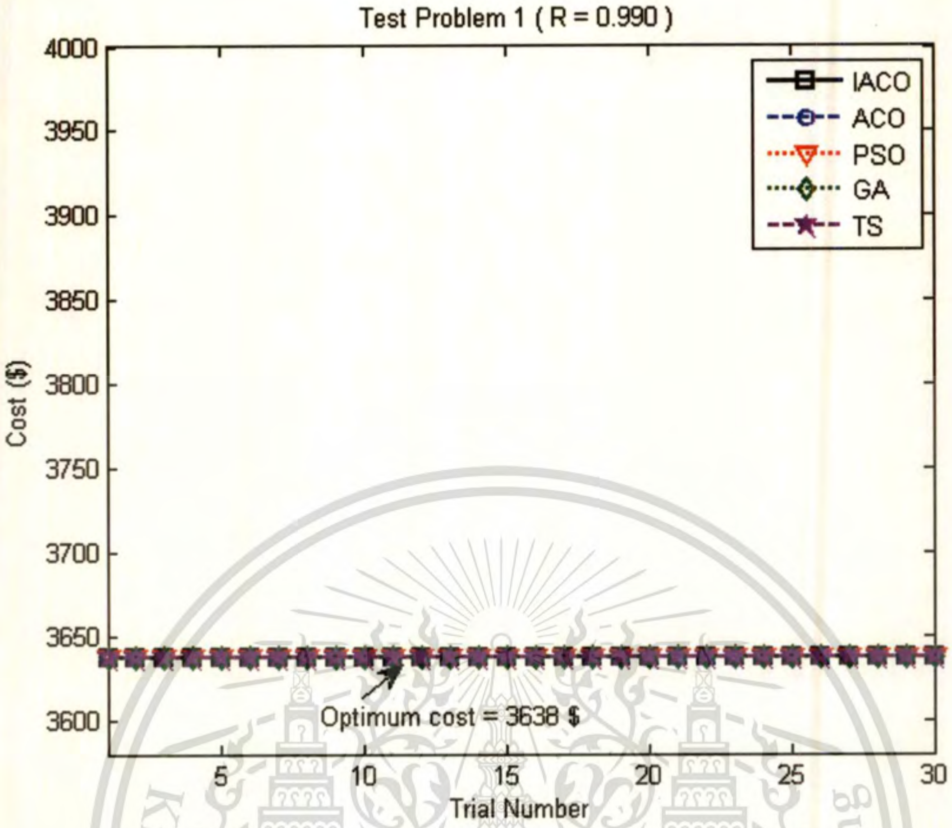


Figure 4.45 Distribution of cost of test problem 1 (R_0 = 0.990)

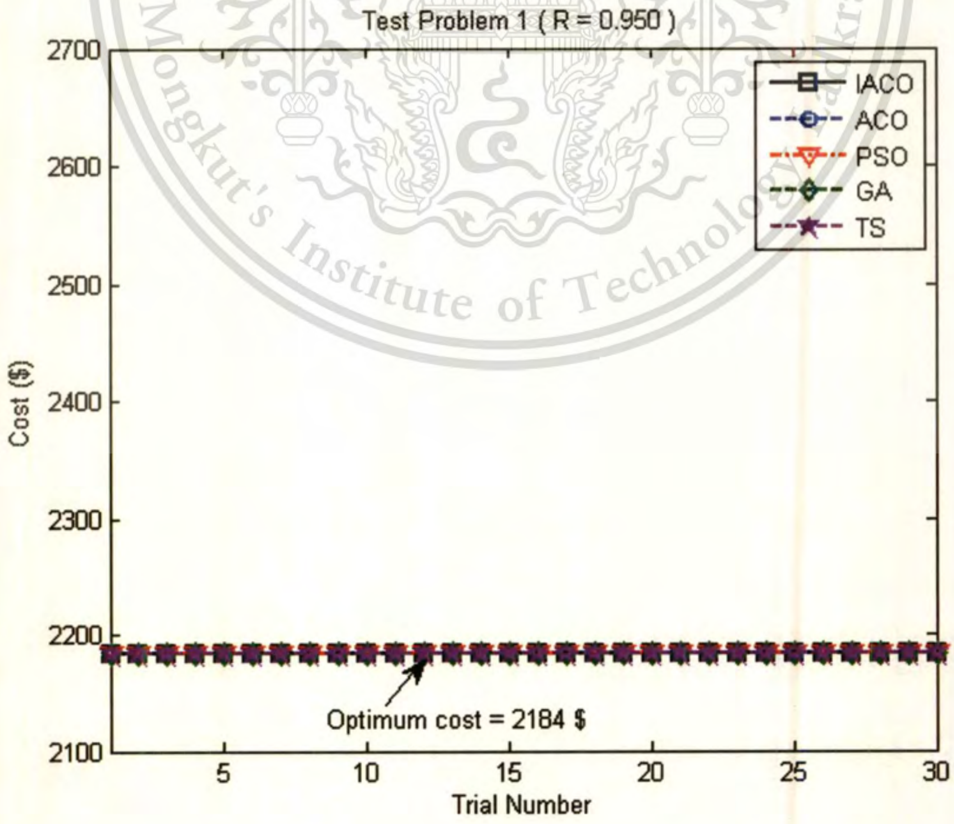


Figure 4.46 Distribution of cost of test problem 1 (R_0 = 0.950)

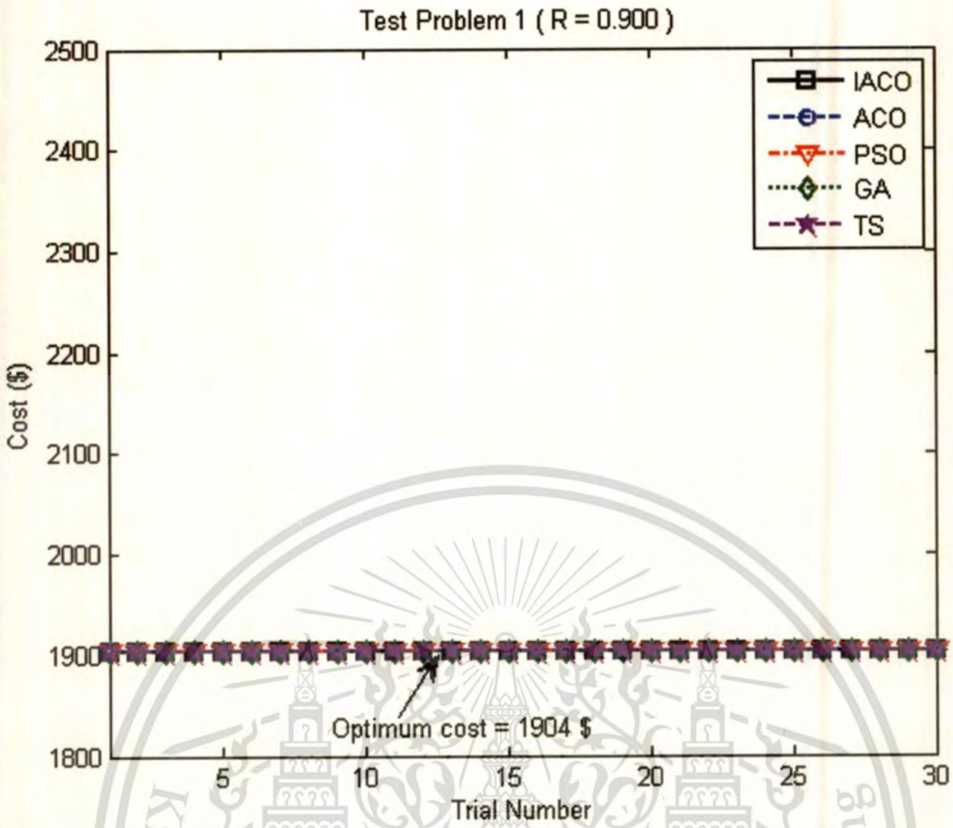


Figure 4.47 Distribution of cost of test problem 1 (R_0 = 0.900)

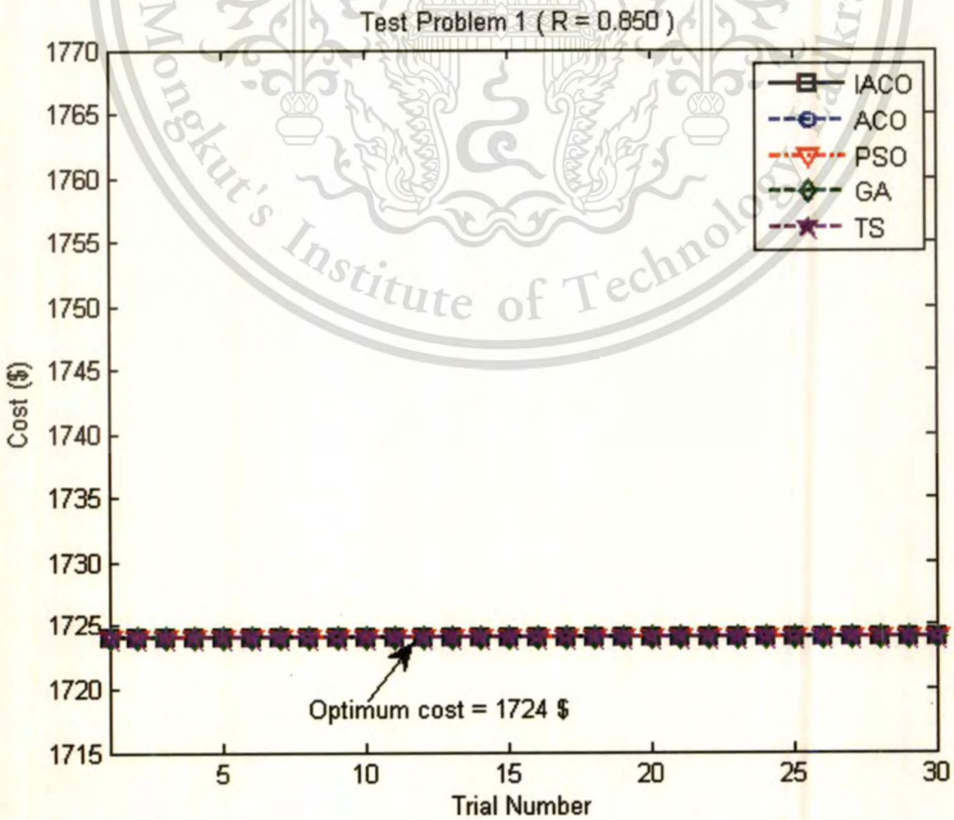


Figure 4.48 Distribution of cost of test problem 1 (R_0 = 0.850)

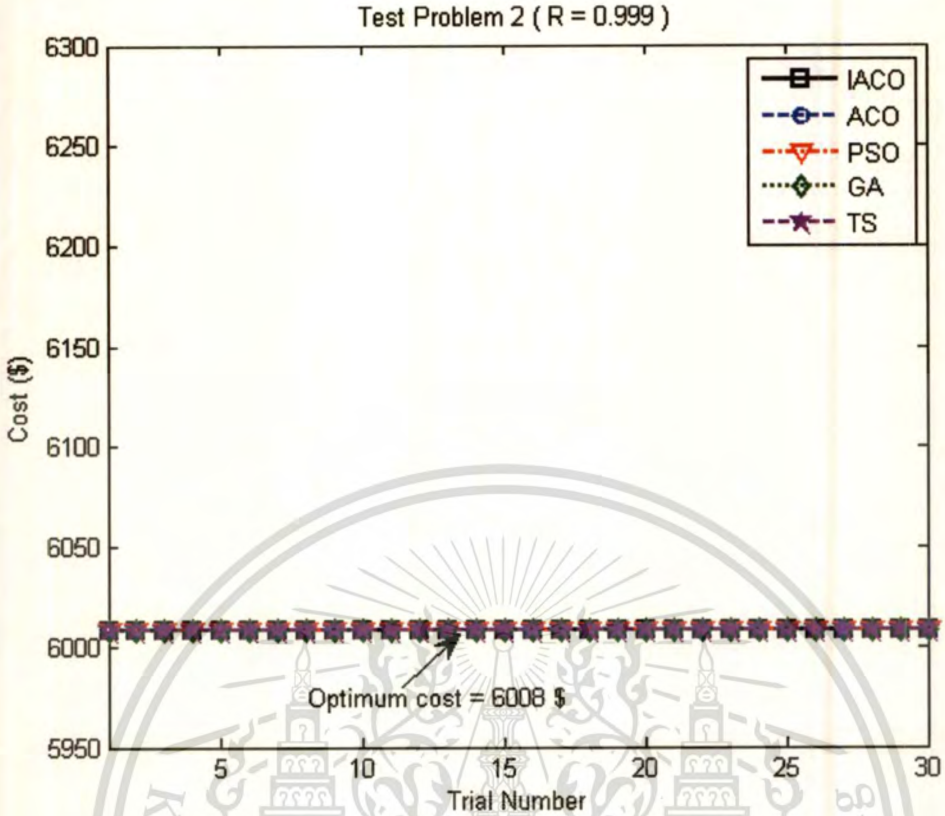


Figure 4.49 Distribution of cost of test problem 2 ($R_0 = 0.999$)

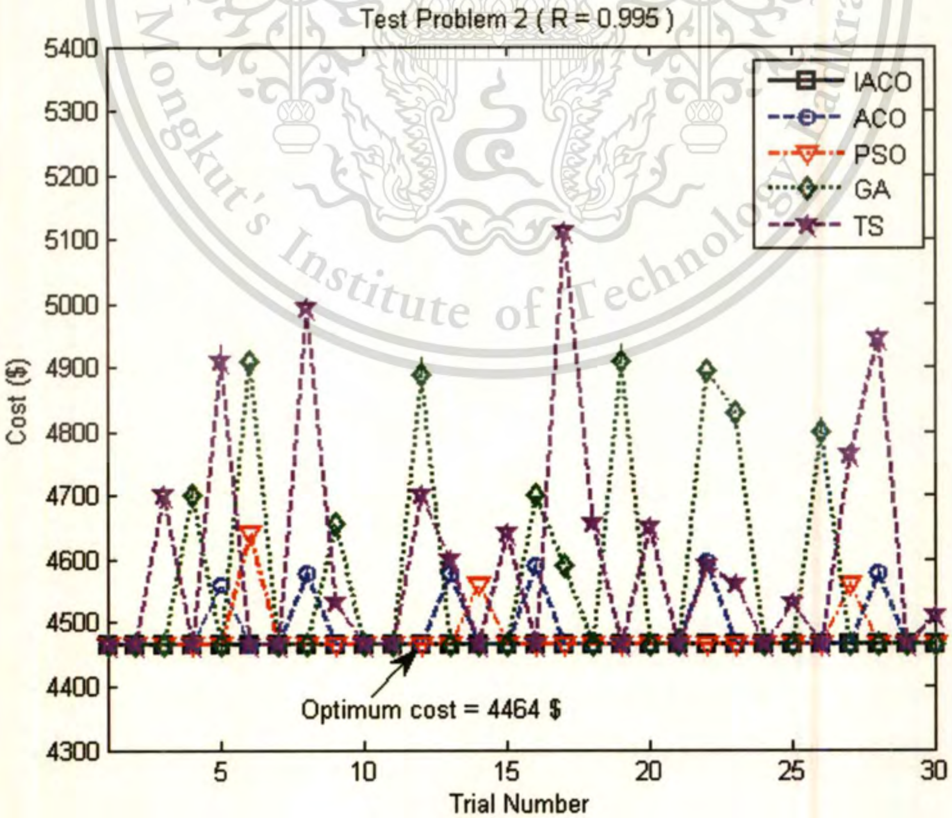


Figure 4.50 Distribution of cost of test problem 2 ($R_0 = 0.995$)

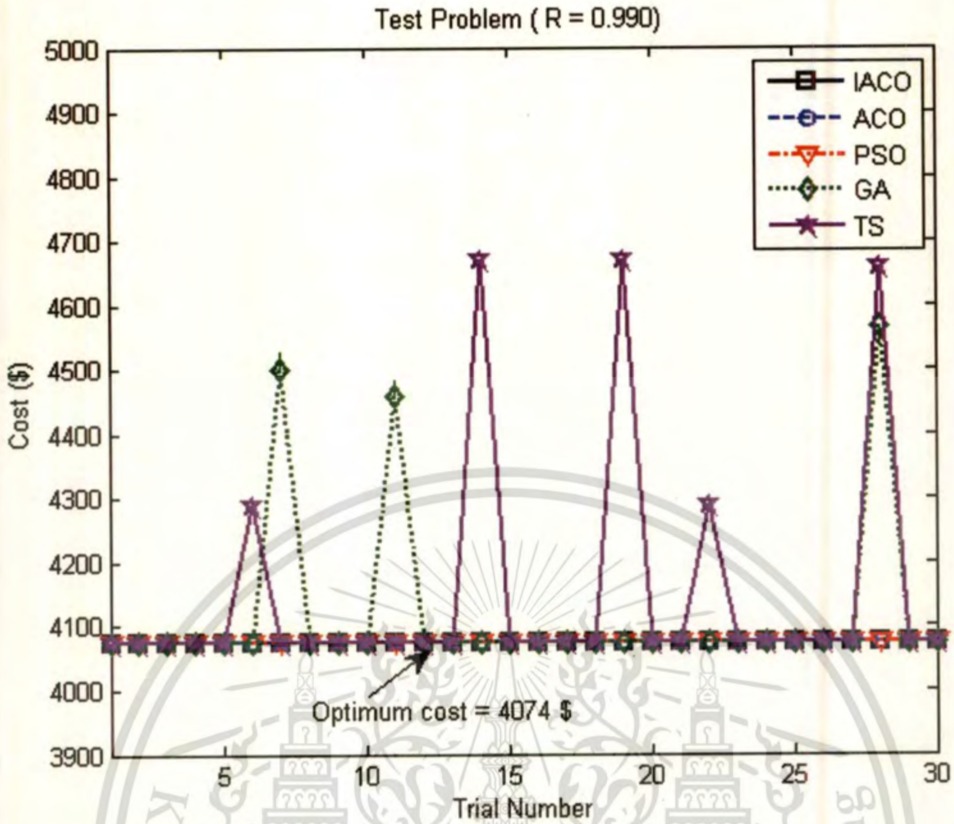


Figure 4.51 Distribution of cost of test problem 2 ($R_0 = 0.990$)

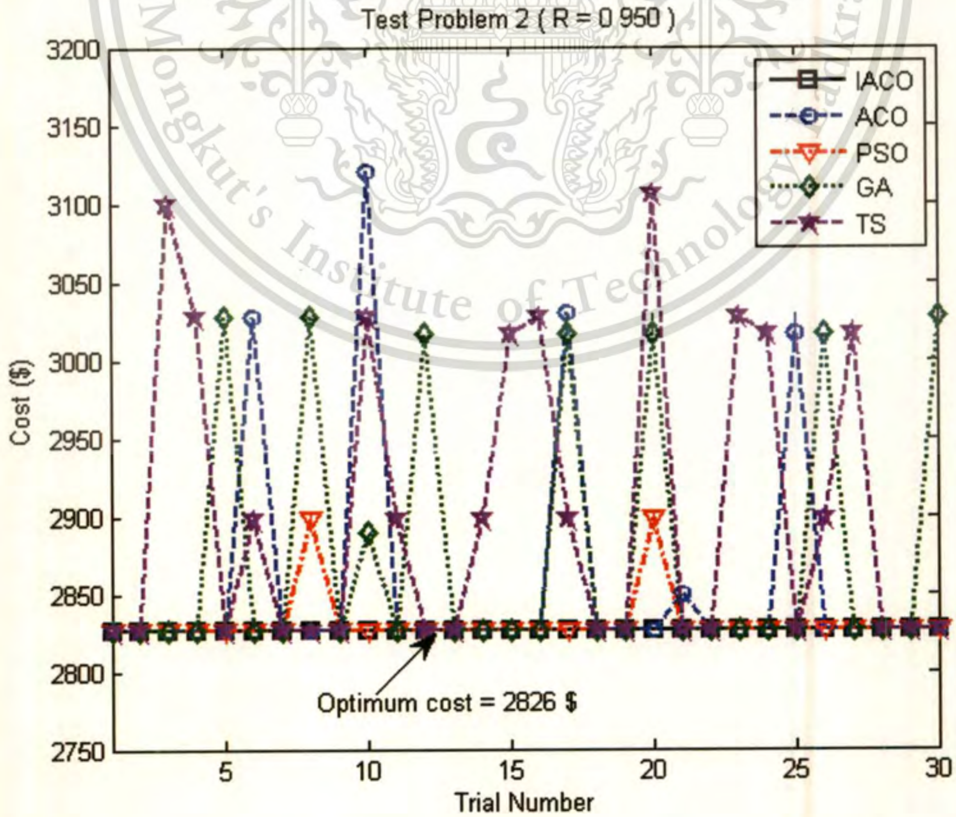


Figure 4.52 Distribution of cost of test problem 2 ($R_0 = 0.950$)

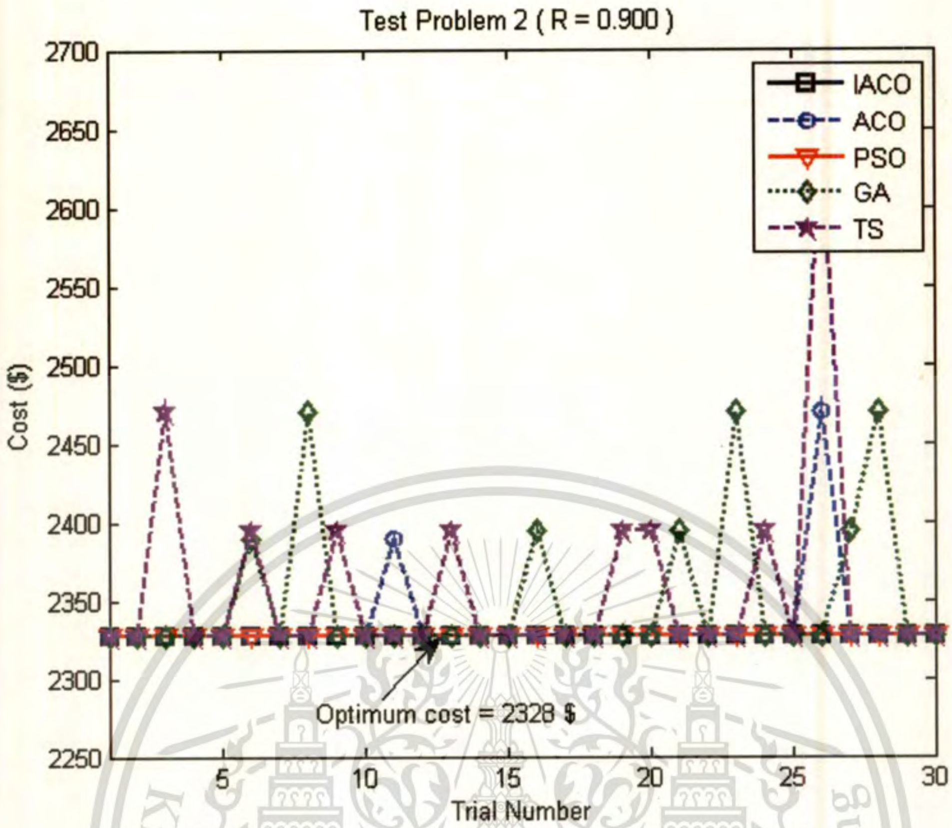


Figure 4.53 Distribution of cost of test problem 2 ($R_0 = 0.900$)

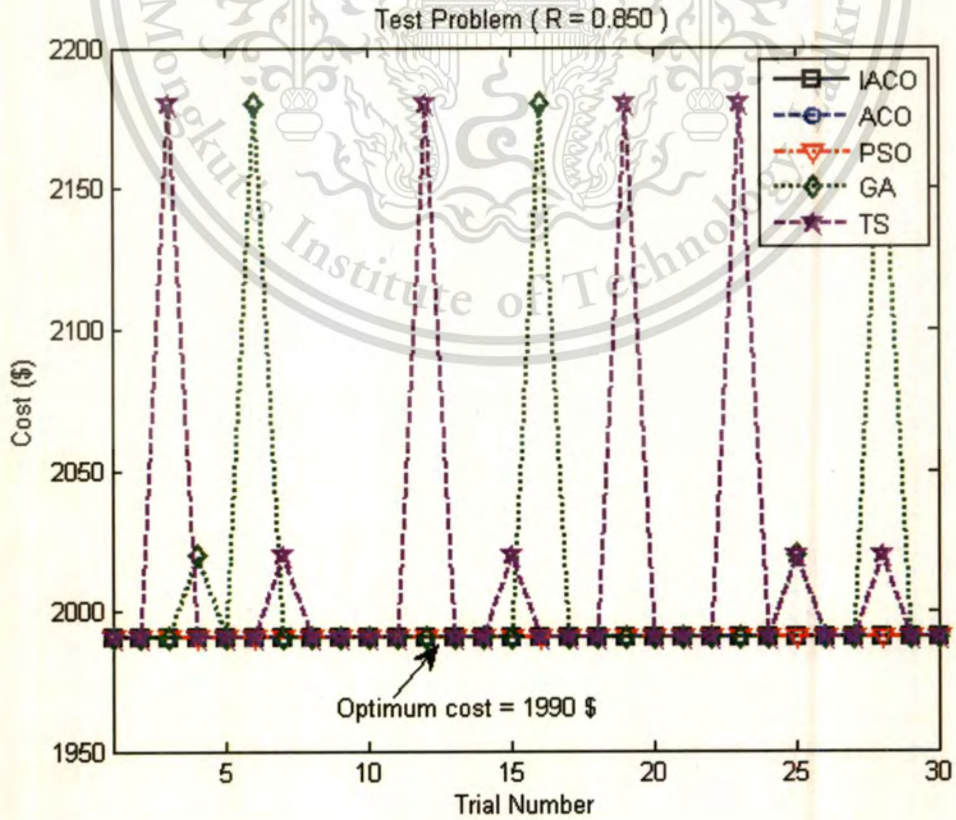


Figure 4.54 Distribution of cost of test problem 2 ($R_0 = 0.850$)

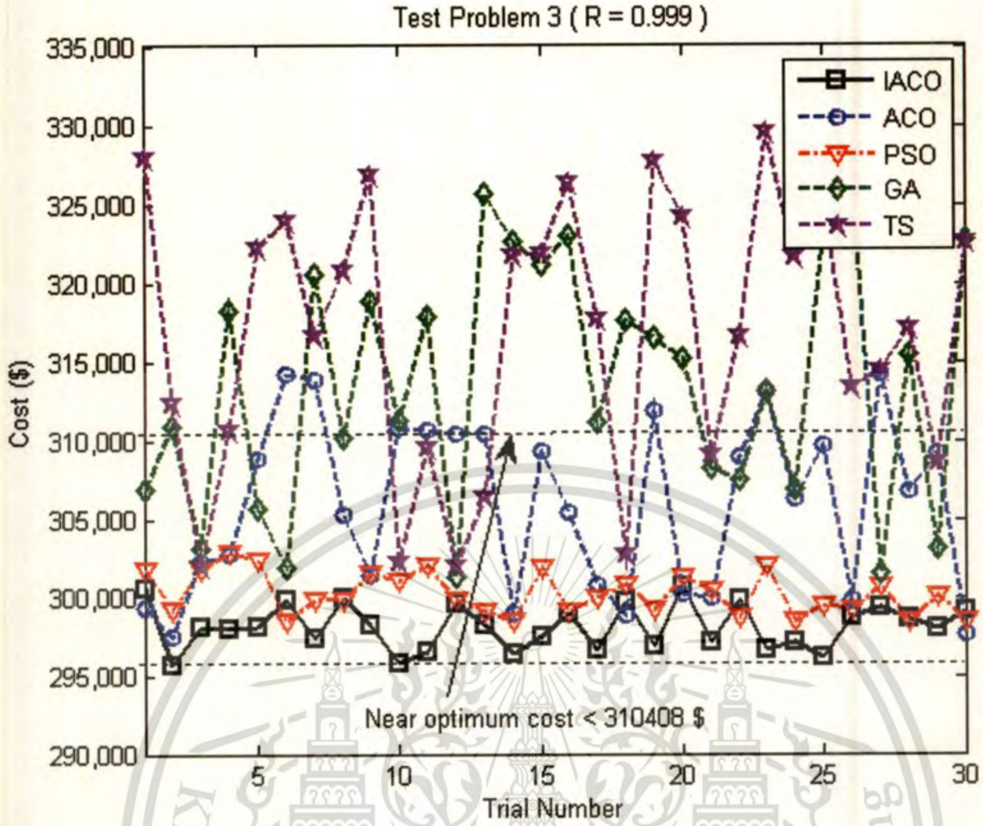


Figure 4.55 Distribution of cost of test problem 3 ($R_0 = 0.999$)

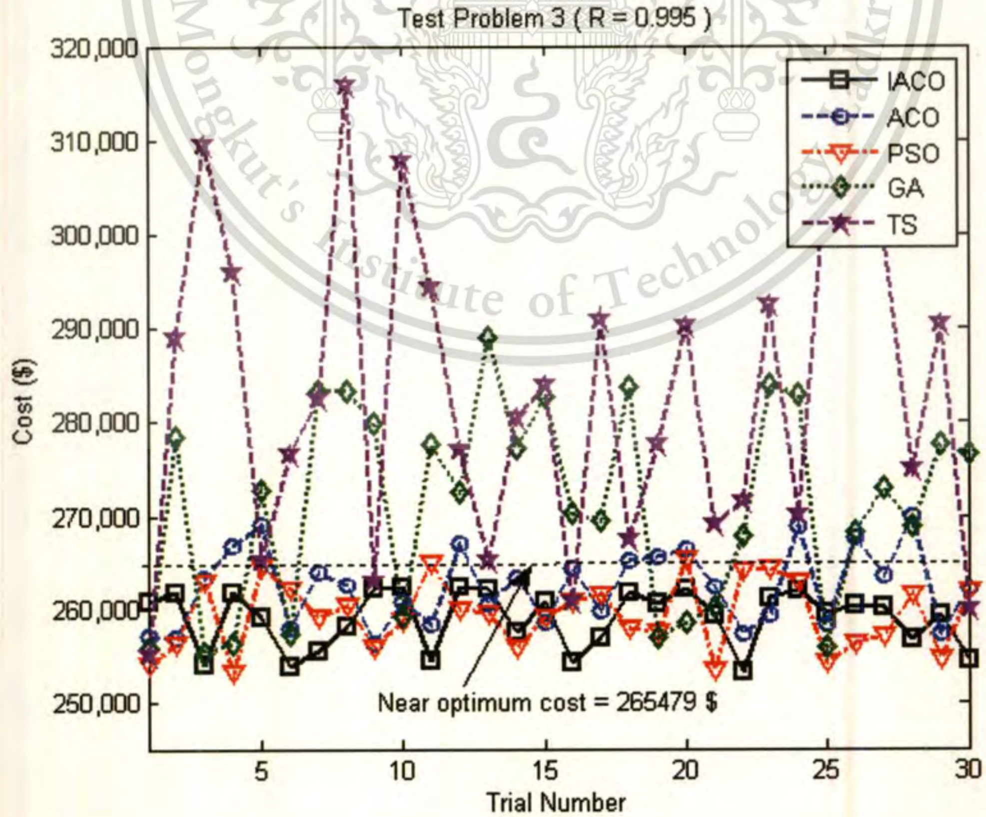


Figure 4.56 Distribution of cost of test problem 3 ($R_0 = 0.995$)

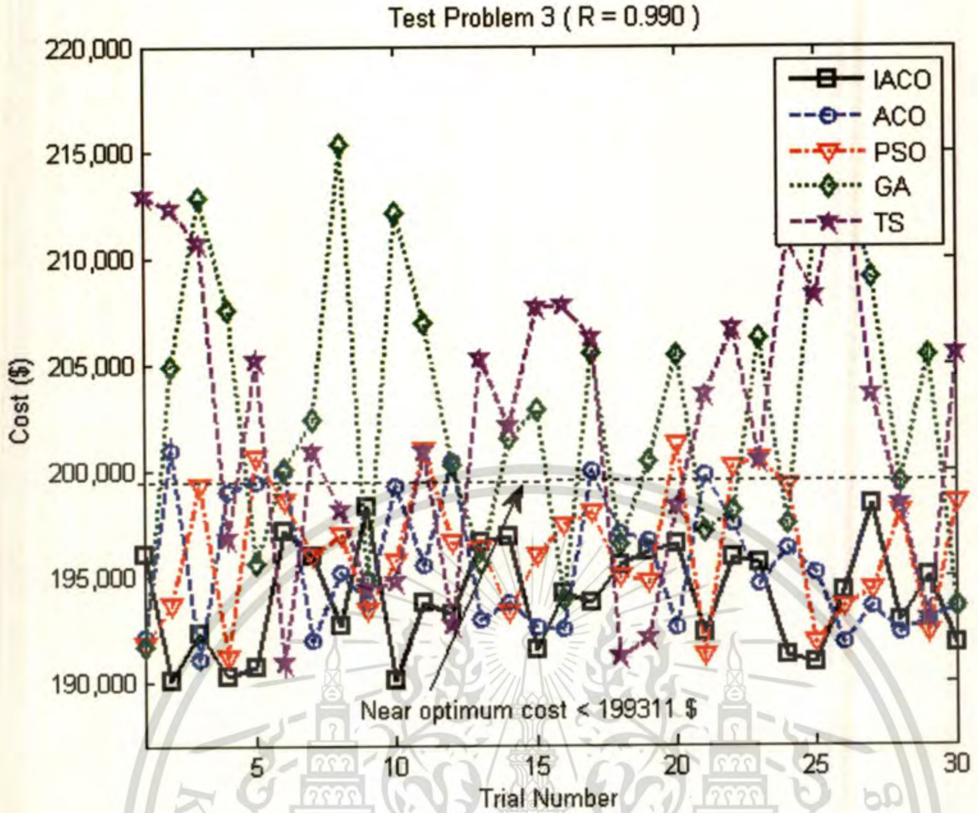


Figure 4.57 Distribution of cost of test problem 3 ($R_0 = 0.990$)

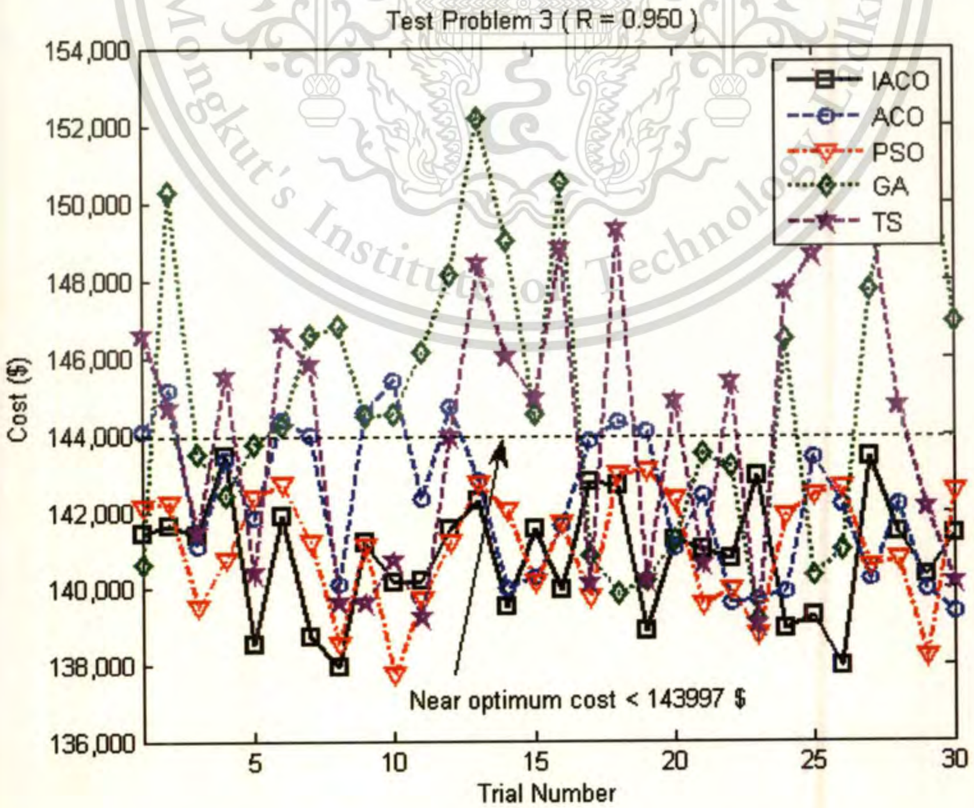
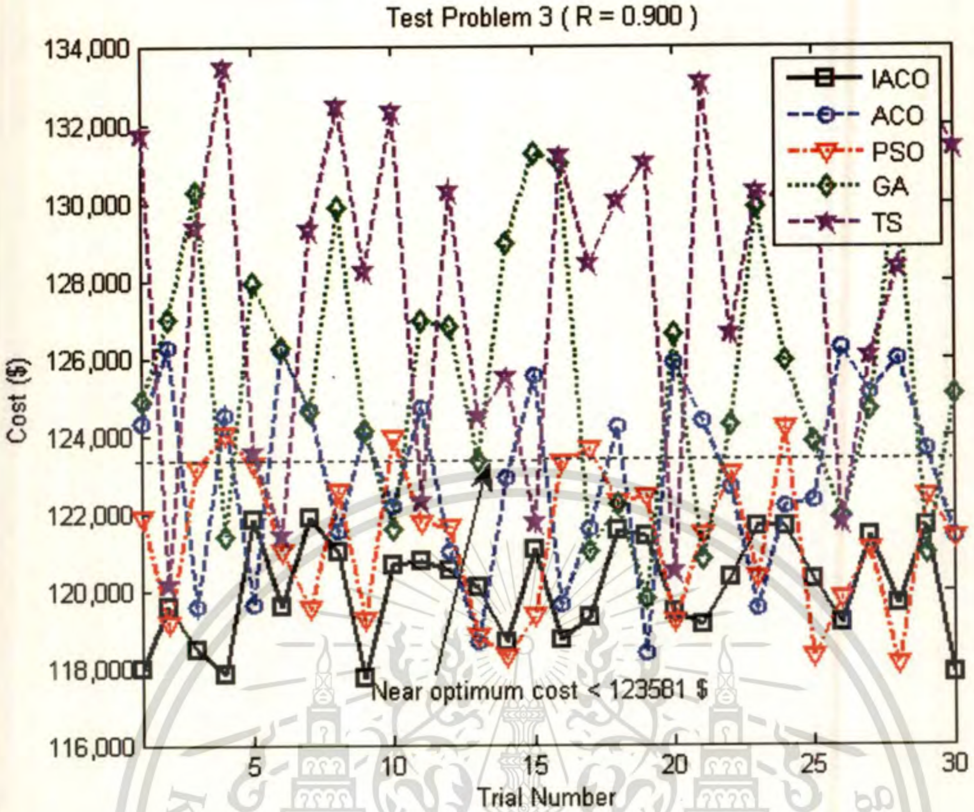


Figure 4.58 Distribution of cost of test problem 3 $R_0 = 0.950$



)

Figure 4.59 Distribution of cost of test problem 3 ($R_0 = 0.900$)

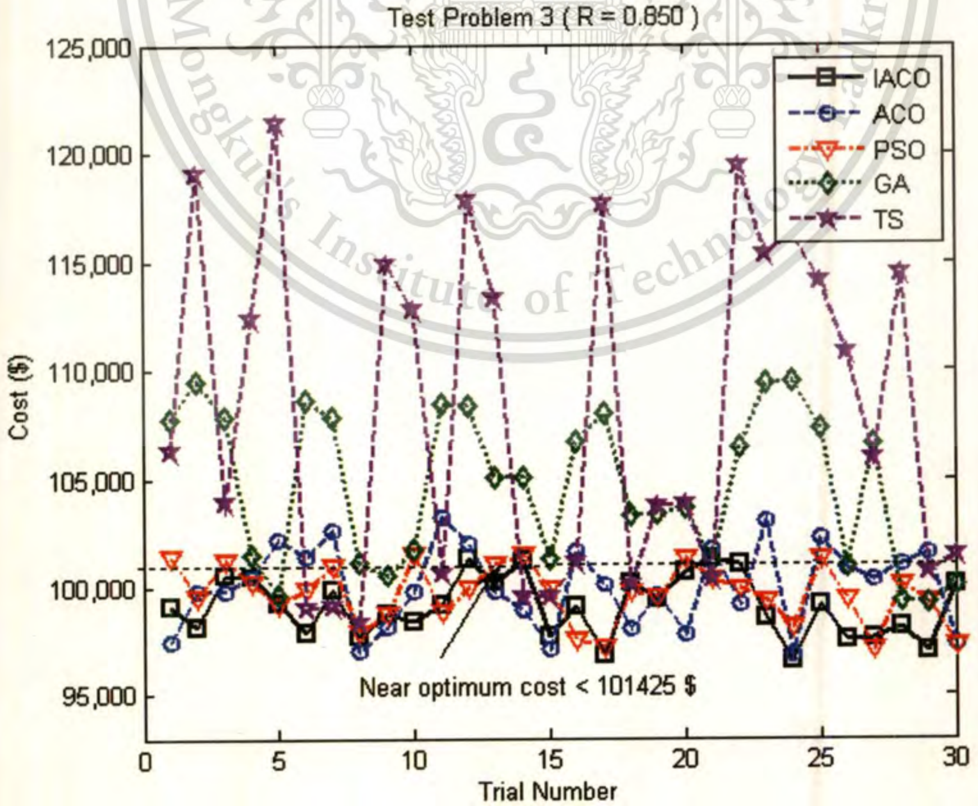


Figure 4.60 Distribution of cost of test problem 3 ($R_0 = 0.850$)

4.4.2 Convergence of Cost

In additional, to depict the efficiency of the proposed method, the convergences of IACO approaches have been compared with others. Figure 4.61 - 4.78 demonstrate the convergence of trials for 6 reliability levels in test problem 1, 2 and 3, respectively. It is found that the convergence of IACO is faster than that of other methods.

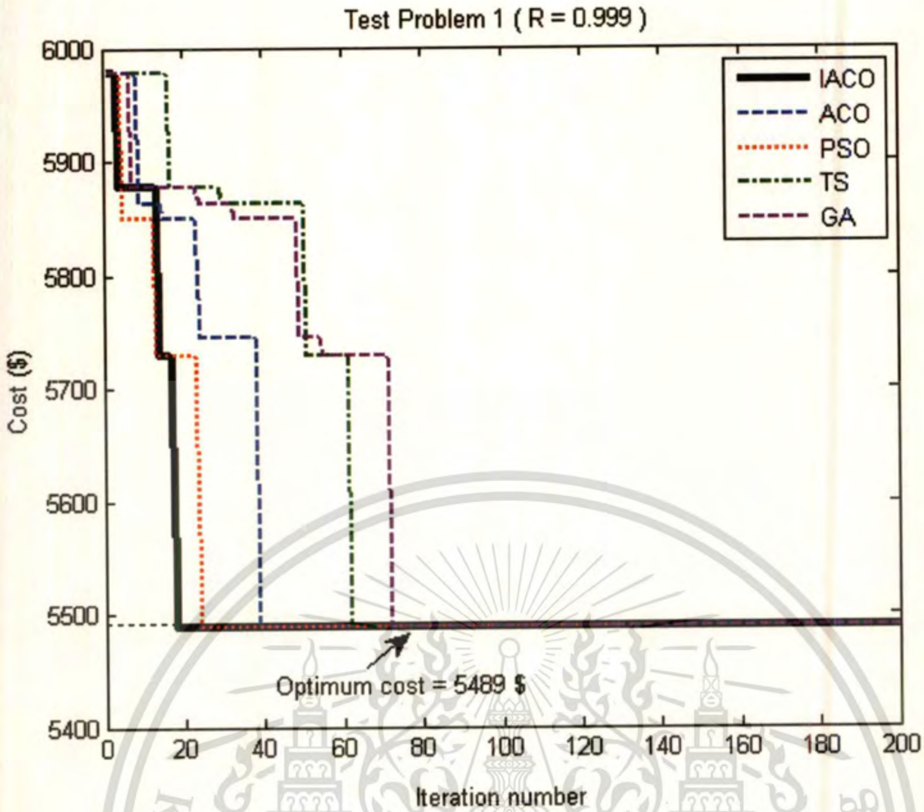


Figure 4.61 Convergence of cost of test problem 1 ($R_0 = 0.999$)

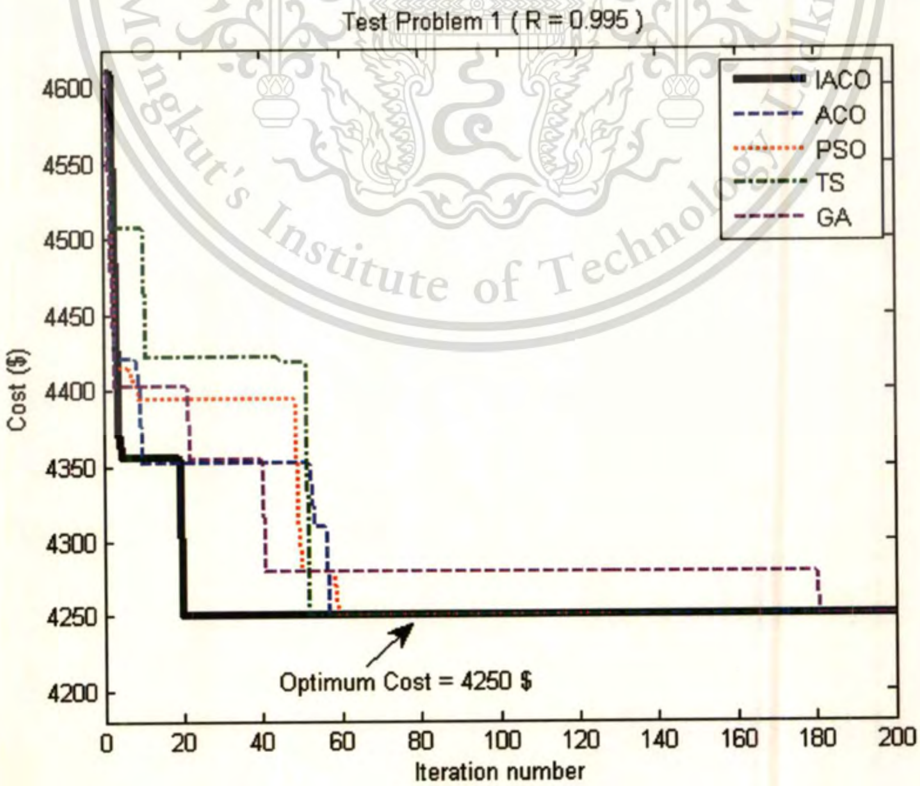


Figure 4.62 Convergence of cost of test problem 1 ($R_0 = 0.995$)

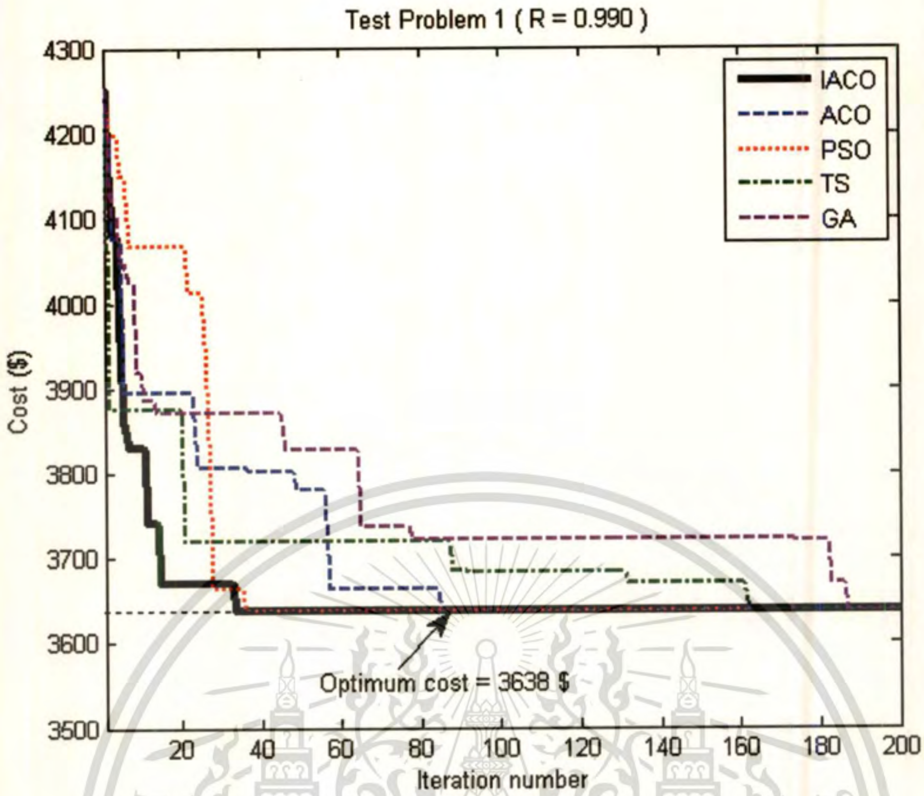


Figure 4.63 Convergence of cost of test problem 1 (R_0 = 0.990)

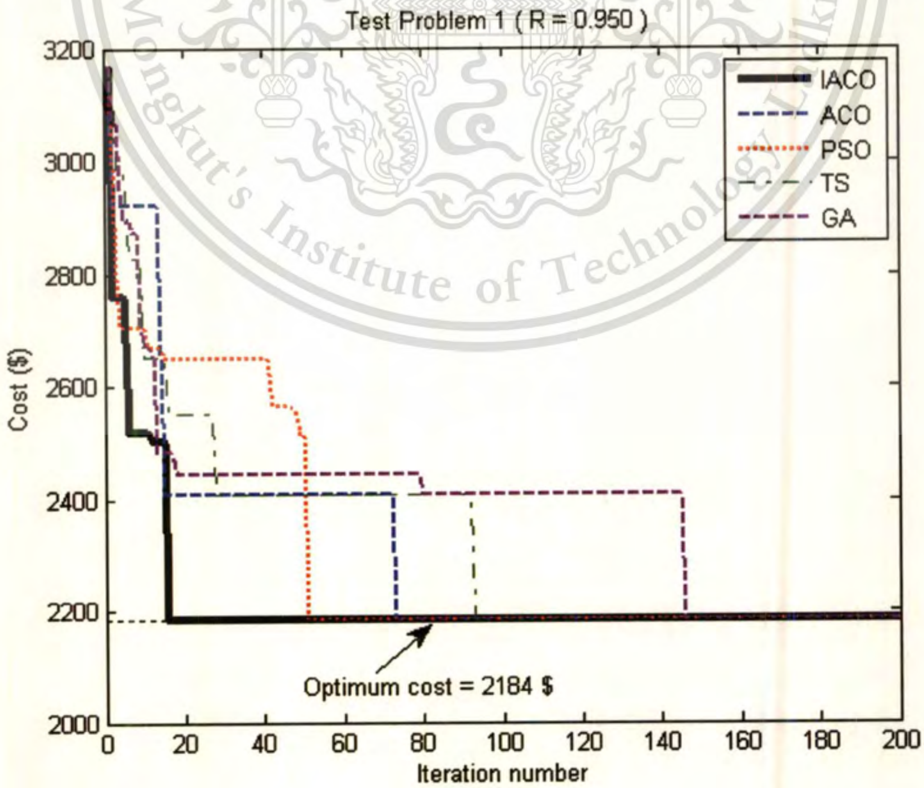


Figure 4.64 Convergence of cost of test problem 1 (R_0 = 0.950)

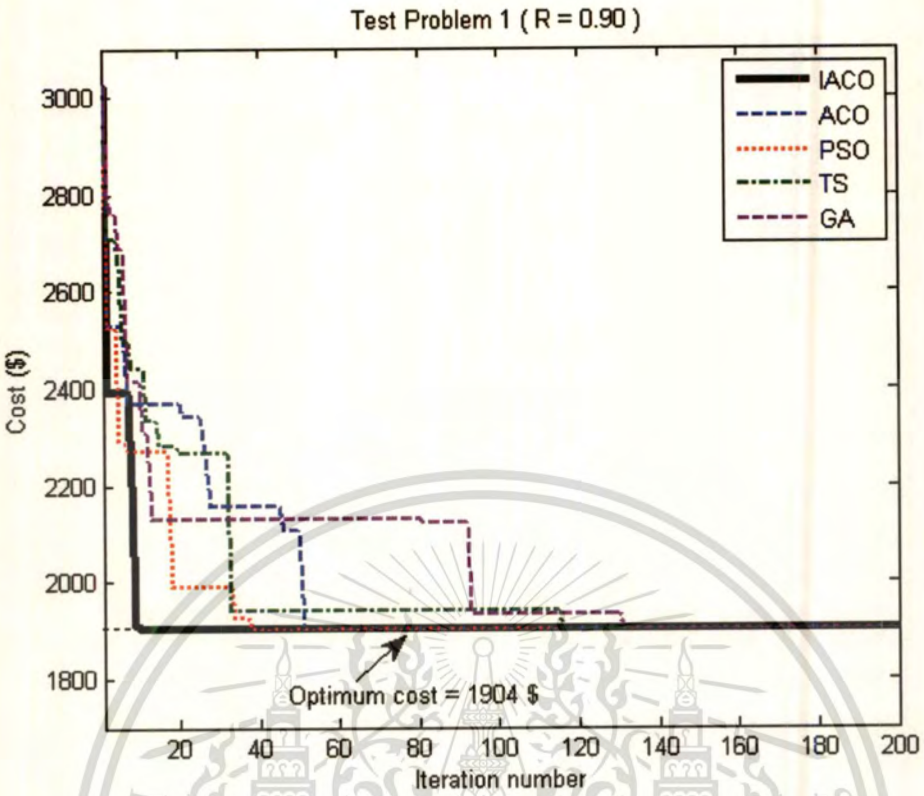


Figure 4.65 Convergence of cost of test problem 1 (R_0 = 0.900)

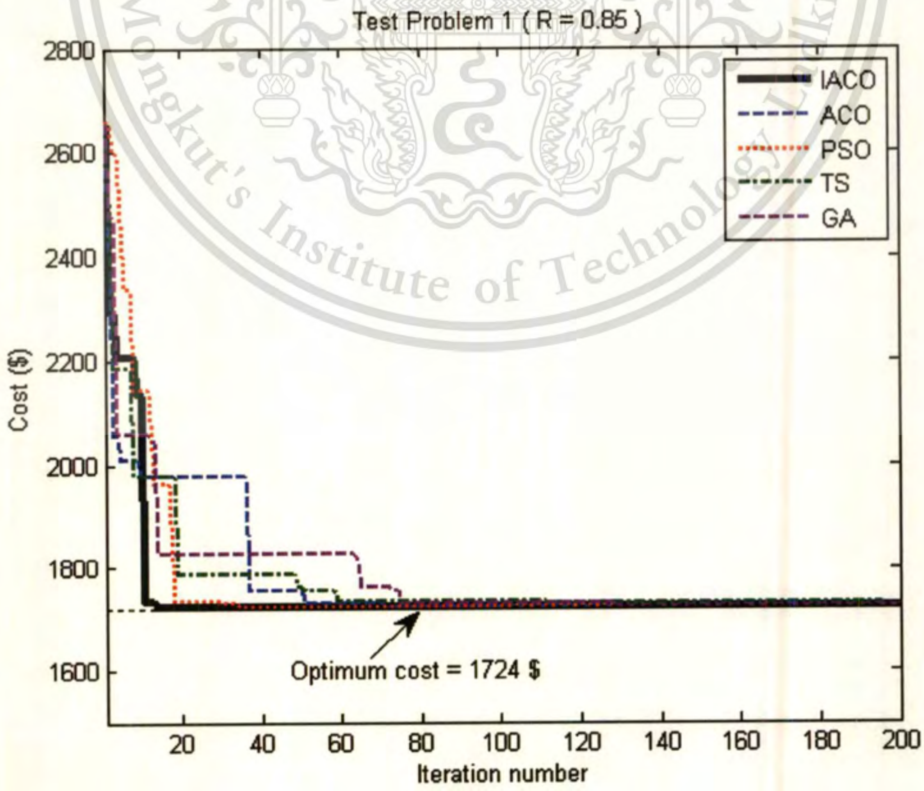


Figure 4.66 Convergence of cost of test problem 1 (R_0 = 0.850)

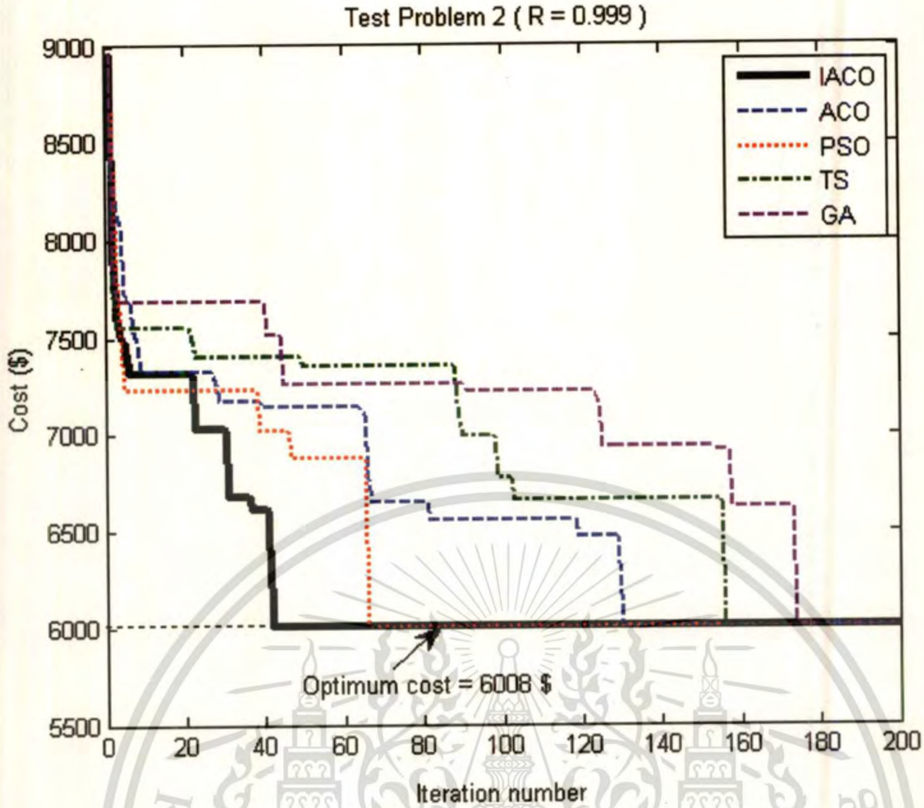


Figure 4.67 Convergence of cost of test problem 2 ($R_0 = 0.999$)

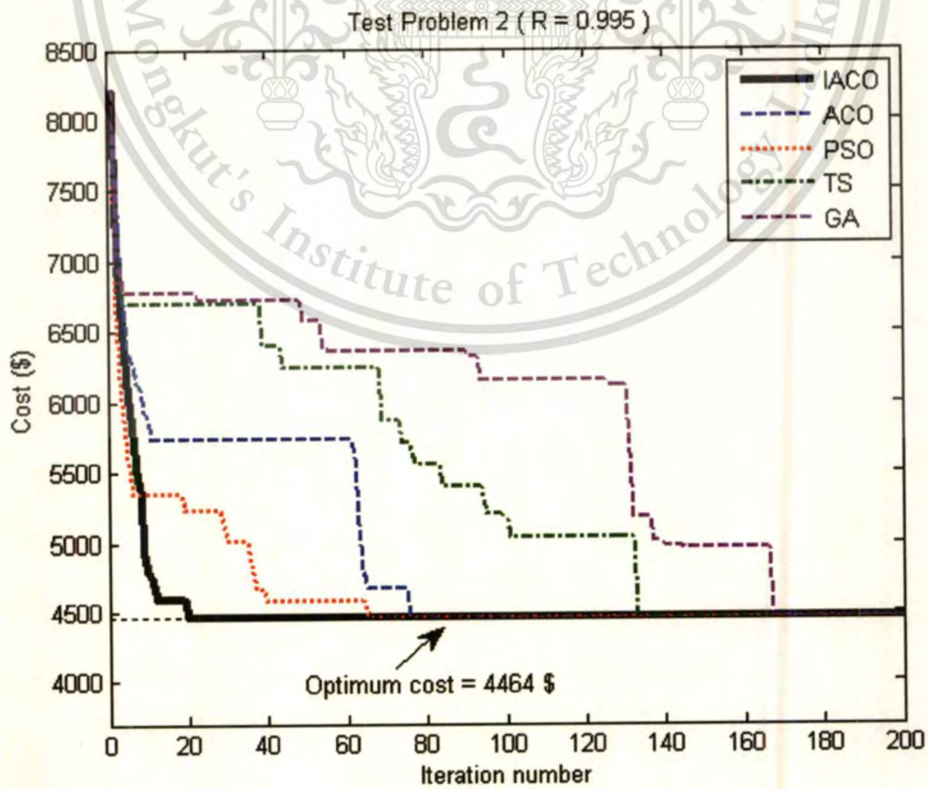


Figure 4.68 Convergence of cost of test problem 2 ($R_0 = 0.995$)

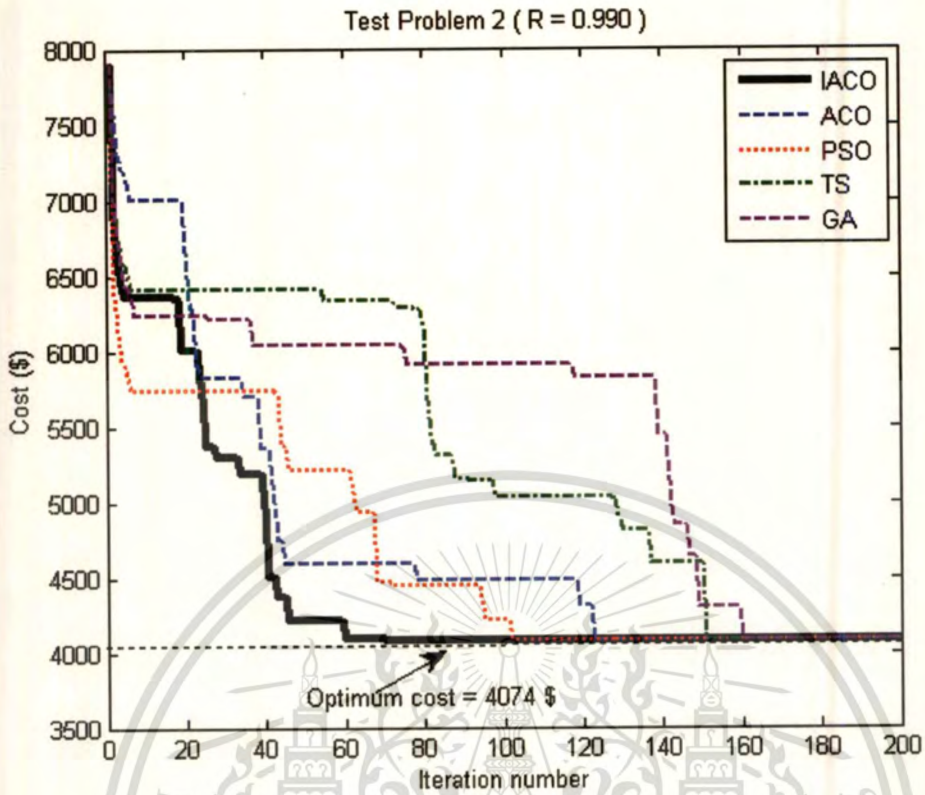


Figure 4.69 Convergence of cost of test problem 2 (R_0 = 0.990)

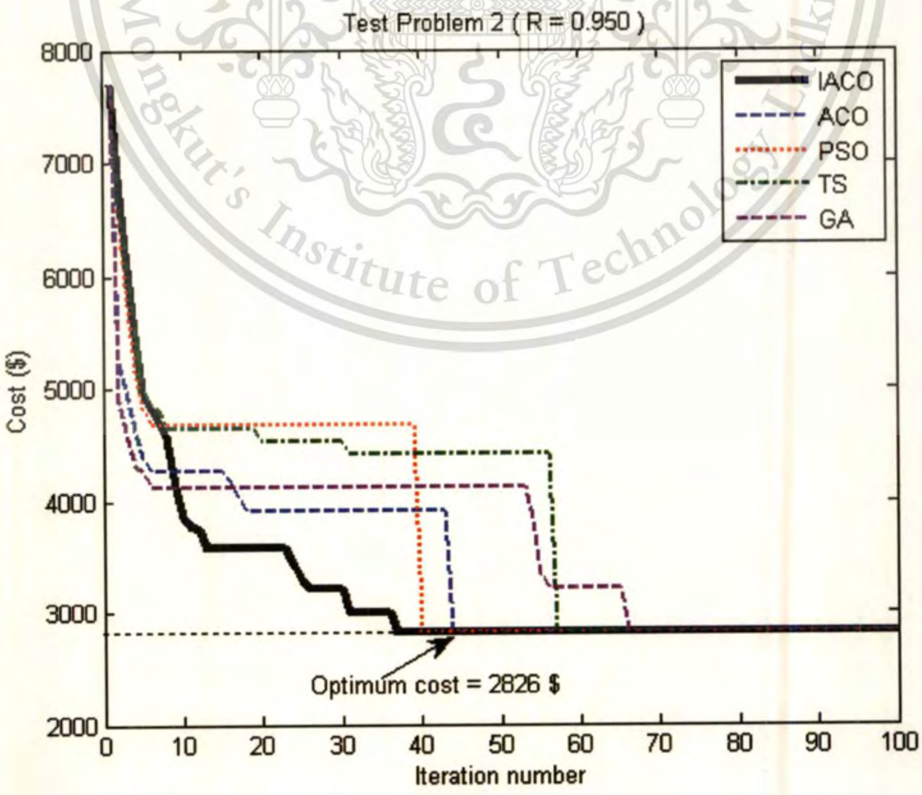


Figure 4.70 Convergence of cost of test problem 2 (R_0 = 0.950)

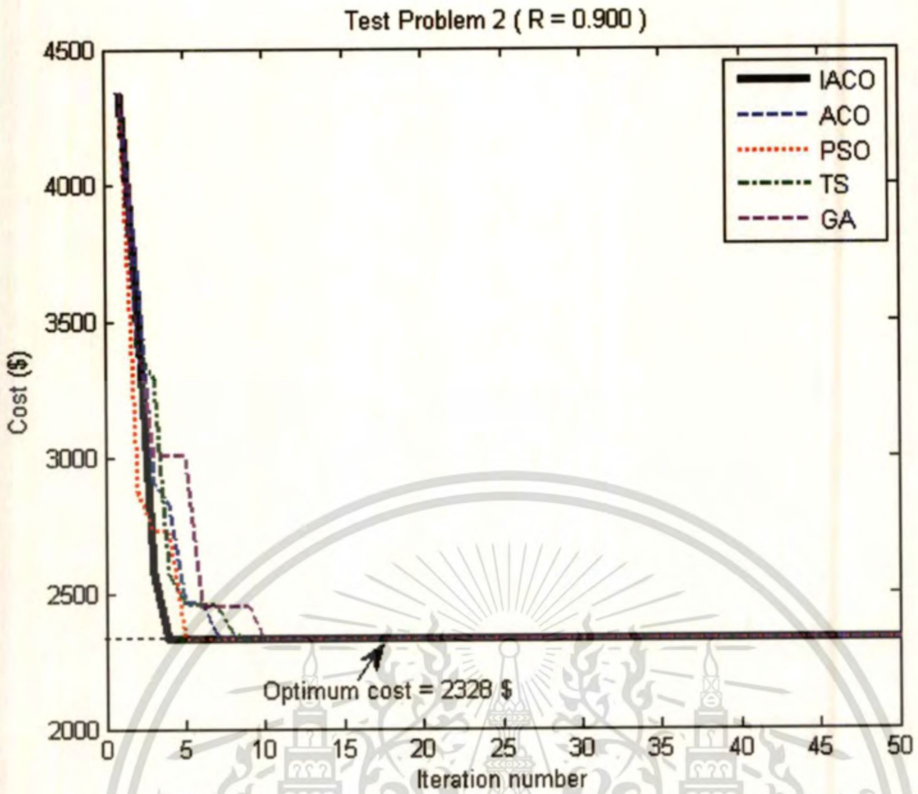


Figure 4.71 Convergence of cost of test problem 2 ($R_0 = 0.900$)

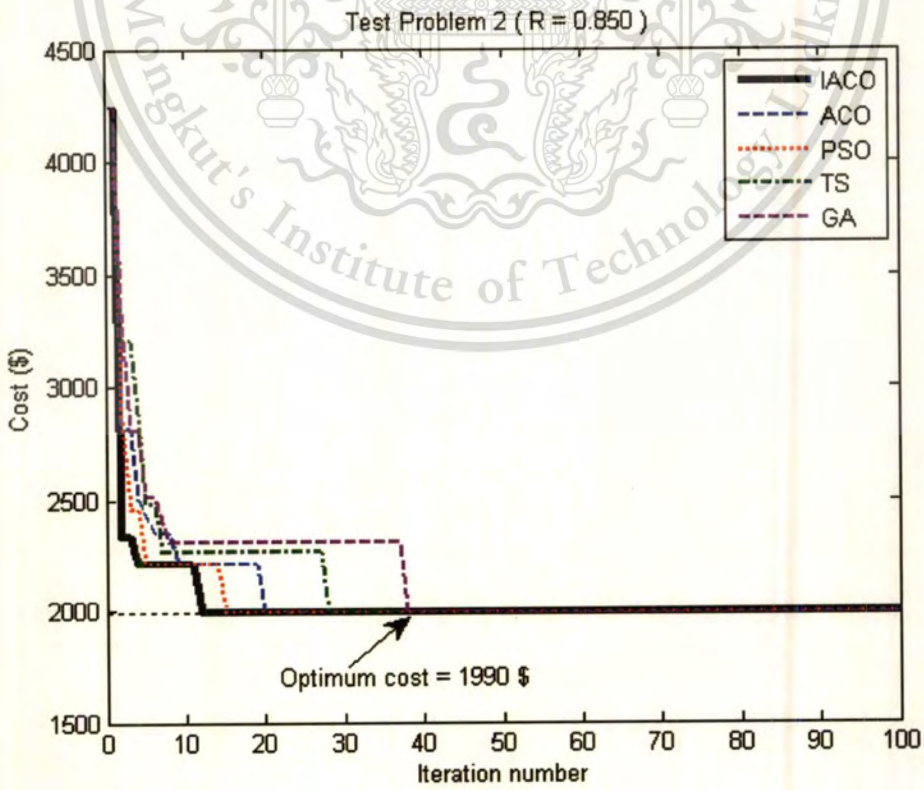


Figure 4.72 Convergence of cost of test problem 2 ($R_0 = 0.850$)

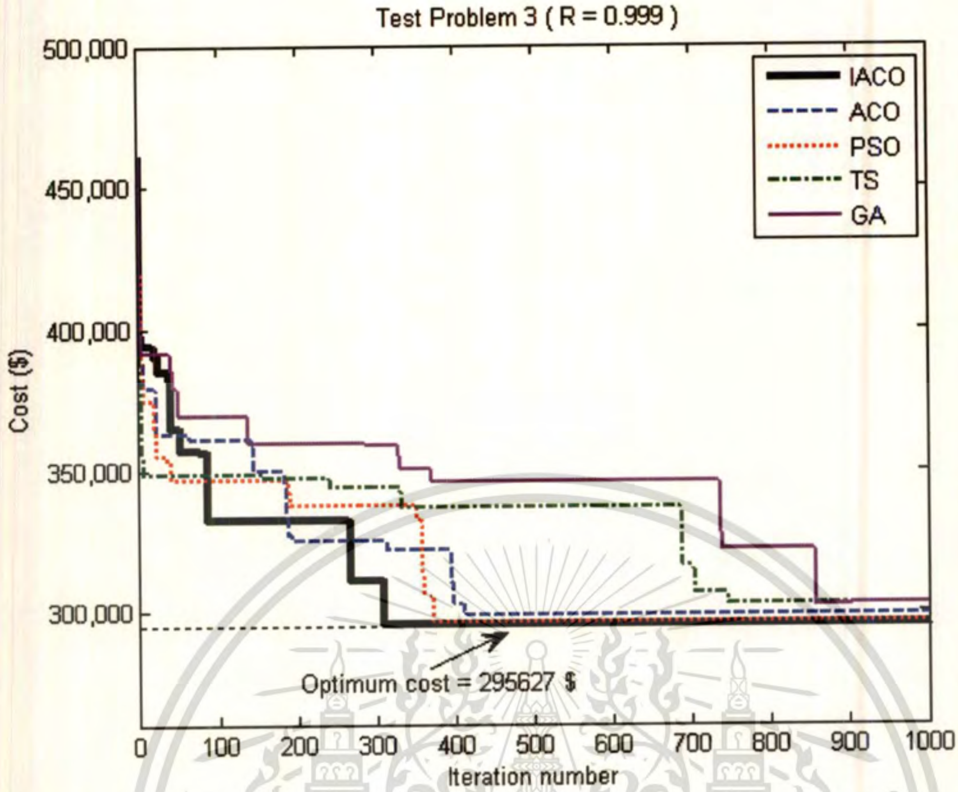


Figure 4.73 Convergence of cost of test problem 3 ($R_0 = 0.999$)

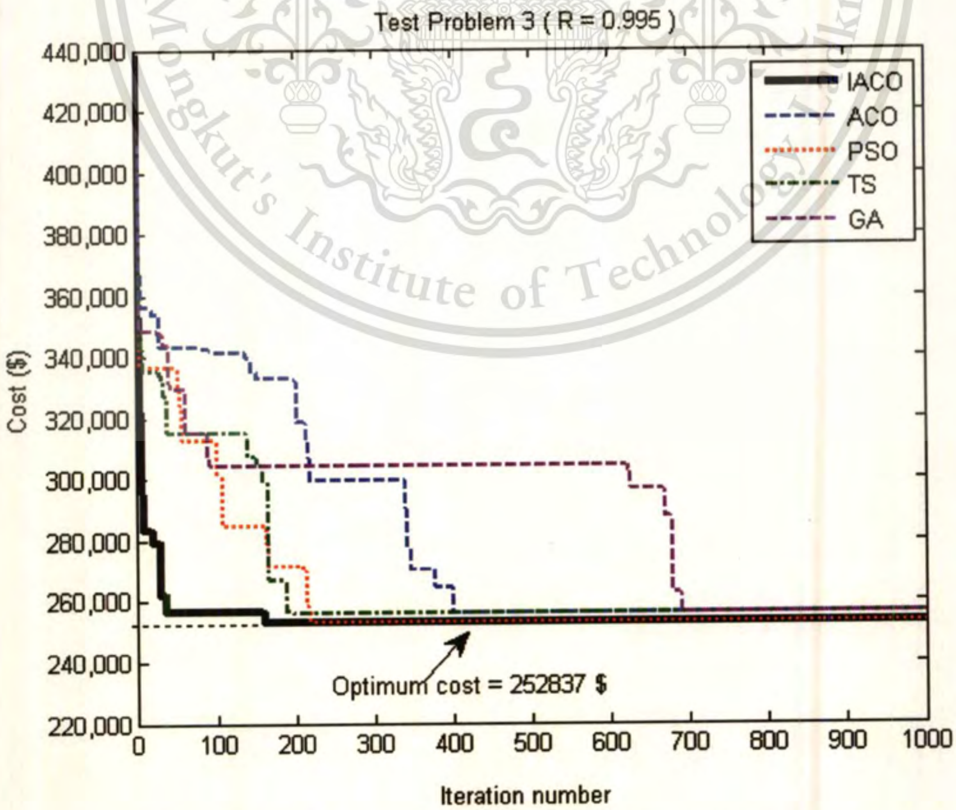


Figure 4.74 Convergence of cost of test problem 3 ($R_0 = 0.995$)

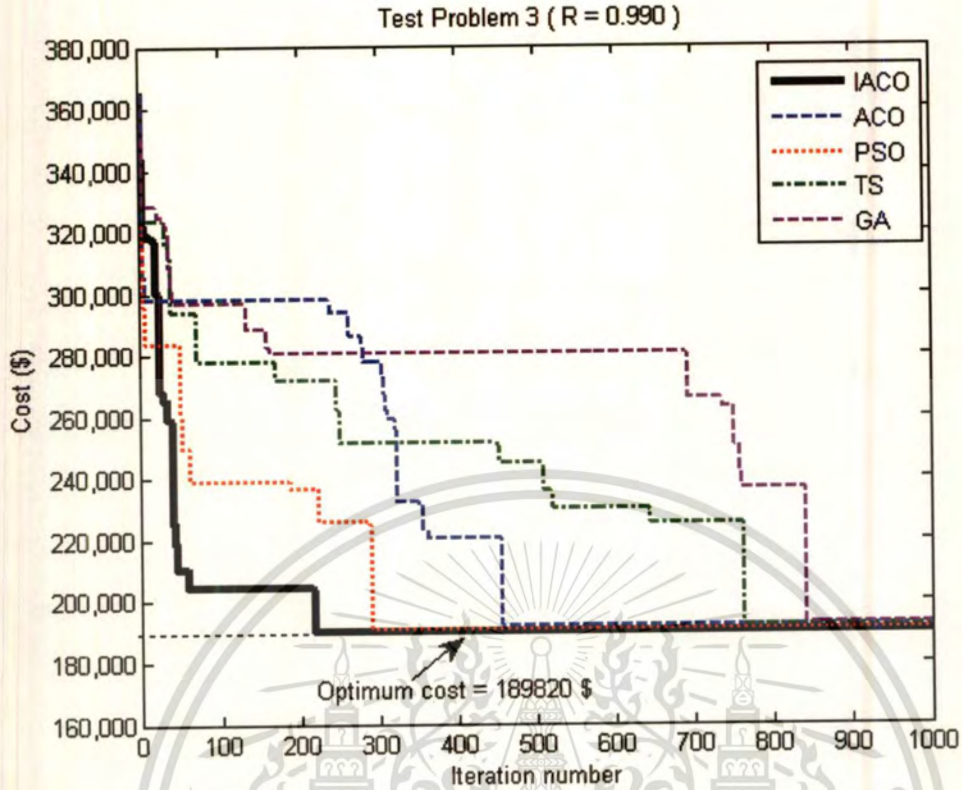


Figure 4.75 Convergence of cost of test problem 3 ($R_0 = 0.990$)

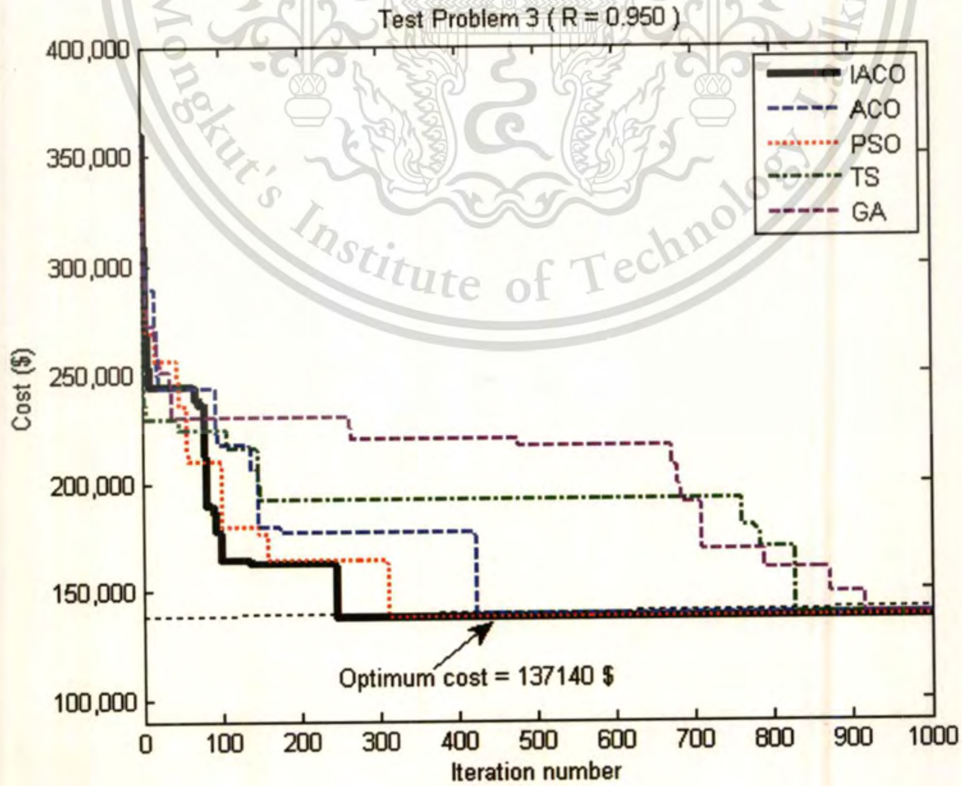


Figure 4.76 Convergence of cost of test problem 3 ($R_0 = 0.950$)

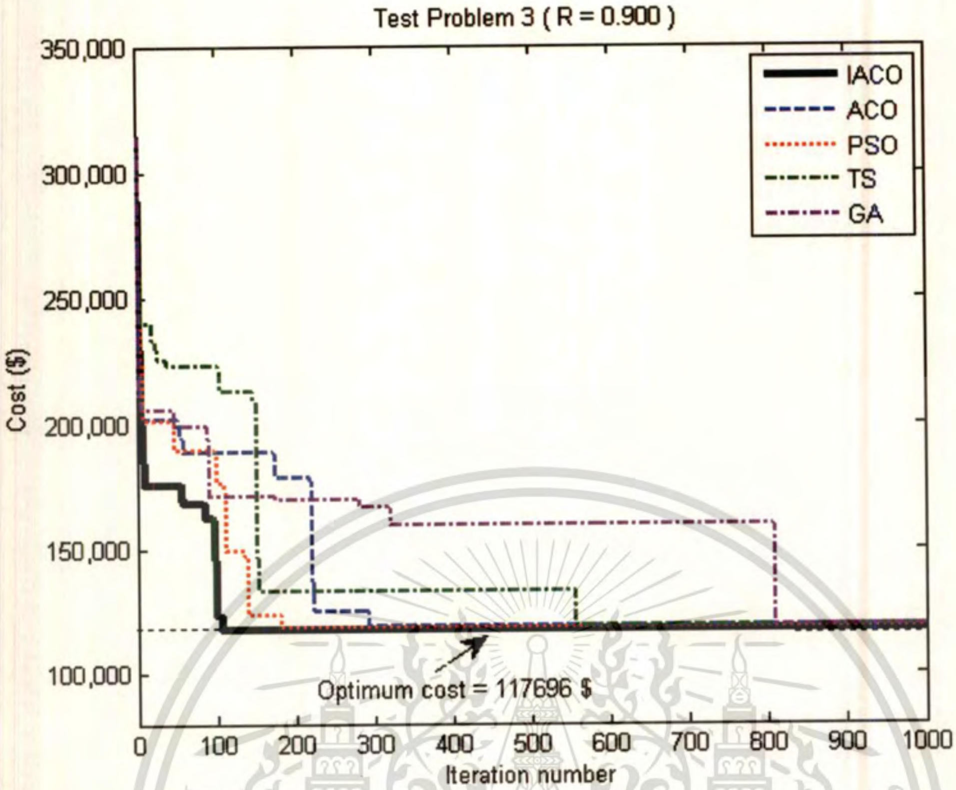


Figure 4.77 Convergence of cost of test problem 3 ($R_0 = 0.900$)

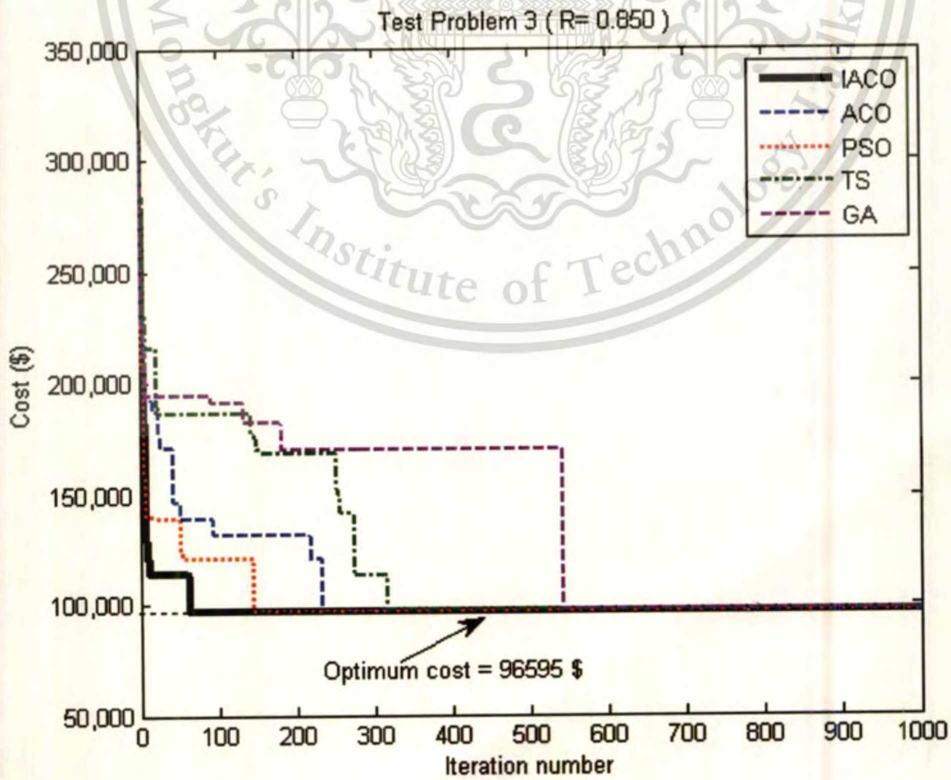


Figure 4.78 Convergence of cost of test problem 3 ($R_0 = 0.850$)

CHAPTER 5

Conclusion

The performance of the improved ACO has been compared with that of differential heuristic algorithm and swarm intelligent techniques i.e. GA, TS, PSO and ACO for multi-dimensional and multimodal numeric problems. Simulation results show that the performance of the IACO algorithm is better than that of the mentioned algorithms. The IACO can be efficiently employed to solve the multimodal engineering problems with high dimensionality.

For telecommunication network design, the proposed IACO algorithm has been implemented to solve the telecommunication network design considering both two-terminal and all-terminal reliability. Numerical testing and a comparative analysis show that the proposed IACO algorithm outperforms other methods i.e. GA, TS, PSO and ACO in terms of high-quality solution, stable convergence characteristic and good computation efficiency.

Referring to simulation results in this thesis, the performance of the proposed IACO algorithm is very good in terms of the local and the global optimizations due to the neighborhood search and re-initialization mechanism used. Consequently, the proposed IACO algorithm is flexible, simple and robust. It can be used efficiently in other optimization problems.

REFERENCES

[1] Rong-Hong Jan, Fung-Jen Hwang, and Sheng-Tzong Chen, **"Topology optimization of a communication network subject to a reliability constraint**," IEEE Transactions on Reliability, Vol.42, No.1, pp.63-70, 1993.

[2] K. K. Aggarwal, Y. C. Chopra, and J.S. Bajwa, **"Reliability evaluation by network decomposition**," IEEE Transactions on Reliability, Vol. R-31, pp.355-358, 1982.

[3] S. Pierre, M. A. Hyppolite, J.M. Bourjolly, and O. Dioume, **"Topology design of computer communication network using simulated annealing**," Eng. Applicat. Artificial Intell., vol.8, pp.61-69, 1995.

[4] A. Kumar, R. M. Pathak, Y. P. Guptaand H. R. Parsaei, **"A genetic algorithm for distributed system topology design**," Comp. Ind. Eng., vol.28, pp. 659-670, 1995.

[5] Darren L. Deeter and Alice E. Smith, **"Economic Design of Reliable Networks,"** IEEE Transactions on Economic of Reliability Engineering, July, 1998.

[6] F. Glover, M. Lee, and J. Ryan, **"Least-cost network topology design for a new service: An application of a tabu search,"** Ann. Oper. Res., vol. 33, pp. 351-362, 1991.

[7] H. F. Beltran and D. Skorin-Kapov, **"On minimum cost isolated failure immune networks,"** Telecomm. Sys, vol. 3, pp. 183-200, 1994.

[8] A. Colorni, M. Dorigo, and V. Maniezzo, **"Distributed optimization by ant colonies,"** in Proc. Eur. Conf. Artificial Life, pp. 134-142, 1991.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, **"Ant system: Optimization by a colony of cooperative agents,"** IEEE Trans. Syst., Man, Cybern. B, Cybern., Vol. 26, no. 1, pp. 29-41, Feb. 1996.

[10] M. Dorigo and LM. Gambardella, **"Ant colonies for the traveling saleman problem,"** Biosystems, vol. 43, pp. 73-81, 1997.

[11] M. Dorigo and LM. Gambardella, **"Ant colony system: a cooperative learning approach to the traveling saleman problem,"** IEEE Trans. Evolut. Comput. vol. 1, no. 1, pp. 53-66, 1997.

[12] LM. Gambardella, E. Taillard, and M. Dorigo, **"Ant colonies for the quadratic assignment problem,"** J. Oper. Res. Soc., vol. 50, pp.167-176, 1999.

[13] J. Bell and P. McMullen, **"Ant colony optimization techniques for the vehicle routing problem,"** Advanced Engineering Informatics, vol. 18, pp. 41-48, 2004.

[14] Holland, J. H. **"Adaptation in Natural and Artificial Systems",** The University of Michigan Press, Ann Arbor, Michigan, 1975.

[15] F. Glover, **"Tabu Search – Part I,"** ORSA Journal on Computing, pp.190-206, 1989.

[16] F. Glover, **"Tabu Search – Part II,"** ORSA Journal on Computing, pp.4-32, 1989

[17] J. Kennedy and R.Eberhart, **"Particle swarm optimization,"** Proc. IEEE International Conf. on Neural Networks (Perth, Australai), IEEE Service Center, Piscataway, NJ, 1995.

[18] G. Flake, **"The Computational Beauty of Nature. Cambridge,"** MA: MIT Press, 1999.

[19] A. Colorni, M. Dorigo, and V. Maniezzo, "**Distributed optimization by ant colonies**," in Proc. Eur. Conf. Artificial Life, pp. 134-142, 1991.

[20] M. Dorigo, V. Maniezzo, and A. Colorni, "**Ant system: Optimization by a colony of cooperative agents**," IEEE Trans. Syst., Man, Cybern. B, Cybern., Vol. 26, no. 1, pp. 29-41, Feb. 1996.

[21] M. Dorigo and LM. Gambardella, "**Ant colonies for the traveling saleman problem**," Biosystems, Vol. 43, pp. 73-81, 1997.

[22] M. Dorigo and LM. Gambardella, "**Ant colony system: a cooperative learning approach to the traveling salesman problem**," IEEE Trans. Evolut. Comput. Vol. 1, no. 1, pp. 53-66, 1997.

[23] Kalinli, A., Karaboga, N., Karaboga, D., 2001. **"A modified touring ant colony optimisation algorithm for continuous functions."** The 16th International Symposium on Computer and Information Sciences (ISCIS XVI), Isık University, Turkey, November 5–7: 437–444.

[24] M. Ball and R. M. V. Slyke, "**Backtracking algorithms for network reliability analysis," Annals of Discrete Mathematics**, Vol. 1, pp. 49-64, 1977.

[25] R. H. Jan, "**Design of reliable networks**," Computers and Operations Research, Vol. 20, no. 1, pp.25-34, 1993.

[26] G. S. Fishman, "**A comparison of four Monte Carlo method for estimating the probability of s-t connectedness**," IEEE Transactions on Reliability, Vol. 35, no.2, pp.145-155, 1986.

[27] R. M. Jose and D. Coit, "**A Monte Carlo Simulation Approach for Approximating Multistate Two-Terminal Reliability**," Reliability Engineering & System Safety, Vol. 87, no. 2, 2005.

[28] S. Shinmori, T. Koide and H. Ishii, "**On lower bound for network reliability by edge-packing**," Transactions of the Japan Society for Industrial and Applied Mathematics, Vol. 5, no. 2, pp. 139-151, 1995.