ขั้นตอนวิธีกลุ่มอนุภาคถ่วงน้ำหนักเฉื่อยแบบปรับค่าได้

นายฐกร ฉัตรชัยสถาพร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

ADAPTIVE INERTIA WEIGHT PARTICLE SWARM ALGORITHM

Mr. Thakorn Chatchaisathaporn

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2012

Thesis Title	ADAPTIVE INERTIA WEIGHT PARTICLE SWARM ALGORITHM
By	Mr. Thakorn Chatchaisathaporn
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Science

(Professor Supot Hannongbua, Dr. rer. nat.)

THESIS COMMITTEE

.....Chairman

(Boonyarit Intiyot, Ph.D.)

.....Thesis Advisor

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

.....Examiner

(Phantipa Thipwiwatpotjana, Ph.D.)

.....External Examiner

(Associate Professor Peerayuth Charnsethikul, Ph.D.)

ฐกร ฉัตรชัยสถาพร : ขั้นตอนวิธีกลุ่มอนุภาคถ่วงน้ำหนักเฉื่อยแบบปรับค่าได้. (ADAPTIVE INERTIA WEIGHT PARTICLE SWARM ALGORITHM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. กรุง สินอภิรมย์สราญ, 47 หน้า.

ปัญหาต้นไม้แบบทอดข้ามสื่อสารเหมาะสุดเป็นปัญหาการหาต้นไม้แบบทอดข้ามที่มีค่าต้นทุนการสื่อสารต่ำที่สุดทั้งยังสอดคล้องกับข้อบังคับของการสื่อสารที่กำหนด เทคนิคที่นิยมใช้ในการแก้ปัญหาดังกล่าวคือใช้ขั้นตอนวิธีทางฮิวริสติก วิธีการทางฮิวริสติกเป็นวิธีที่ให้ผลเฉลยที่ดีภายในเวลาการประมวลผลที่สมเหตุสมผล ขั้นตอนวิธีการหาค่าเหมาะสุดฝูงอนุภาค เป็นขั้นตอนวิธีทางฮิวริสติกวิธีหนึ่งในการแก้ปัญหาการหาค่าเหมาะสุด ในงานนี้เราขยายหลักการของขั้นตอนวิธีการหาค่าเหมาะสุดฝูงอนุภาคสำหรับปัญหาต้นไม้แบบทอดข้ามสื่อสารเหมาะสุด ซึ่งนำเสนอโดย Hoang และคณะ โดยรวมแนวความคิดกลยุทธ์ค่าน้ำหนักเฉื่อยแบบปรับค่าได้ใช้กับขั้นการปรับปรุงความเร็ว เราสรุปผลกระทบของค่าน้ำหนักเฉื่อยแบบปรับค่าได้ ของขั้นตอนวิธีที่นำเสนอ นอกจากนี้เรายังแนะนำการเริ่มต้นประชากรแบบใหม่ ขั้นตอนวิธีที่ปรับปรุงแล้วของเราให้ผลเฉลยที่มีคุณภาพดี

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา คณิตศาสตร์ประยุกต์และวิทยาการคณนา ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2555 .....

5373805623 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : COMMUNICATION NETWORKS / NETWORK DESIGN AND COMMUNICATION / PARTICLE SWARM OPTIMIZATION-BASED / ADAPTIVE INERTIA WEIGHT / INITIAL POPULATION

THAKORN CHATCHAISATHAPORN : ADAPTIVE INERTIA WEIGHT PARTICLE SWARM ALGORITHM. ADVISOR : ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., 47 pp.

An Optimum Communication Spanning Tree (OCST) problem is a problem of finding a spanning tree of minimum total communication cost satisfying a given set of requirements of communication. The popular technique for solving OCST problem is to use the heuristic algorithm. The heuristic approach does successfully obtain good solutions in a reasonable computational time. The particle swarm optimization-based (PSO) algorithm is one of the heuristic algorithms for optimization problems. In this work, we extend the concept of the particle swarm optimization-based (PSO) algorithm for the OCST problem proposed by Hoang et al. by combining the concept of adaptive inertia weight strategy to the velocity update step. We summarize the effect of the adaptive inertia weight over the proposed algorithm. In addition, we also introduce a new pattern of population initialization. Our proposed algorithm yields a better solution quality.

Department : Mathematics and Computer Science Student's Signature .....
Field of Study : Applied Mathematics and Computational Science Advisor's Signature .....
Academic Year : 2012

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I would like to thank the department of Mathematics and Computer Science of Chulalongkorn University for giving me the opportunity to pursue and finally finish this master education. I would like to express my deepest appreciation to my advisor, Asst. Prof. Dr. Krung Sinapiromsaran, for his excellent supervision, constant support, and encouraging me in every way and every step to complete my thesis.

I am indebted to the thesis committee, Dr. Boonyarit Intiyot, Dr. Phantipa Thipwiwatpotjana, and Assoc. Prof. Dr. Peerayuth Charnsethikul, for their contributions and invaluable advice. Special thanks to Prof. Suchada Siripant, who had given me invaluable experiences of broadening my vision in education and work, and Prof. Dr. Gerhard Reinelt, who introduced me to the OCST problem and gave me helpful comments during my research period in the Discrete & Combinatorial Optimization research group at University of Heidelberg. My thanks are extended to all members in the research group especially to Stefan Wiesberg, Jens Fielenbach, and Stefan Lörwald for their generous advice and guidance in programming implementation. Special thanks to Tuan Nam Nguyen for helping me contact Vietnamese authors and forwarding me a collection of benchmark instances. My master thesis could not be accomplished without them. I really appreciate them all.

I would also like to thank Erasmus Mundus Mobility with Asia (EMMA) scholarship for giving me an opportunity to do research in Germany for nine months. Another group of persons I cannot fail to mention is the group of Thai students and EMMA beneficiaries in Heidelberg for their hospitality, generosity, and support, especially to Nopporn Thamrongrat, Dr. Ratinan Boonklurb, Dr. Apichat Suratanee. All of them made my stay in Heidelberg an unforgettable moment of my life. I am very grateful to know them all.

I would like to thank all my friends and colleagues in the AMCS program especially to Wacharasak Siriseriwan, Wasakorn Laesanklang, Charoenchai Sirisomboonrat, Panote Songwattanasiri, Suebkul Kanchanasuk, Aua-aree Boonperm, and the AMCS'53 group for their useful advice, companionship, and encouragement.

Last but not least, I would give a big thanks and love to my parents, Surachai and Jidapa Chatchaisathaporn, for giving me unconditional love and continuous support.

CONTENTS

Page

ABSTRACT IN THAI iv

ABSTRACT IN ENGLISH v

ACKNOWLEDGEMENTS vi

CONTENTS vii

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER I INTRODUCTION 1

1.1 Objective 3

1.2 Thesis overview 3

CHAPTER II LITERATURE REVIEW 4

2.1 Optimum communication spanning tree problem 4

2.2 Particle swarm optimization 10

2.3 Particle swarm optimization with adaptive inertia weight 19

CHAPTER III ADAPTIVE INERTIA WEIGHT PARTICLE SWARM ALGORITHM FOR THE OCST PROBLEM 22

3.1 Population initialization 22

3.2 Velocity initialization 23

3.3 The measure of dissimilarity between a pair of trees 23

3.4 Adaptive inertia weight 25

3.5 Flowchart 25

CHAPTER IV EXPERIMENTAL RESULTS 27

CHAPTER V CONCLUSION 39

REFERENCES 40

APPENDIX 43

BIOGRAPHY 47

LIST OF TABLES

Page

Table 1: The obtained solution (Total Communication Cost) for the OCST benchmark instances 29

Table 2: The percentage of gap between the obtained solution and the best known solution 29

LIST OF FIGURES

Page

Figure 2-1: Diagram represents the Palmer6 network 5

Figure 2-2: Diagram represents a spanning tree T extracted from the Palmer6 network 6

Figure 2-3: Diagram represents the Palmer6 network with weights 14

Figure 2-4: Diagram shows Step 1 of the decode phase for the Palmer6 network 14

Figure 2-5: Diagram shows Step 2 of the decode phase for the Palmer6 network 15

Figure 2-6: Flowchart shows the PSO algorithm 18

Figure 3-1: T_{k_1}, a tree extracted from the Palmer6 network 24

Figure 3-2: T_{k_2}, a tree extracted from the Palmer6 network 24

Figure 3-3: Flowchart shows our AIW-PSO algorithm for the OCST problem 26

Figure 4-1: The best obtained solutions using four algorithms for Palmer24 31

Figure 4-2: The best obtained solutions using four algorithms for Raidl50 31

Figure 4-3: The best obtained solutions using Li&Bouchebaba's GA and our AIW-PSO for Berry35u 32

Figure 4-4: Comparison graph shows obtained solutions of Palmer24 over iterations between O-PSO and AIW-PSO 33

Figure 4-5: Comparison graph shows obtained solutions of Raidl50 over iterations between O-PSO and AIW-PSO 33

Figure 4-6: Comparison graph shows obtained solutions of Raidl75 over iterations between O-PSO and AIW-PSO 34

Figure 4-7: The obtained solutions from various population size settings of Palmer24 35

Figure 4-8: The obtained solutions from various population size settings of Raidl75 35

Figure 4-9: The obtained solution of Palmer24 over iterations with reduced population size 36

Figure 4-10: The obtained solution of Raidl50 over iterations with reduced population size 36

Figure 4-11: The average value of obtained solutions of Palmer24 from PSO with Hoang's initial pattern and with new initial pattern when population size is 50 37

Chapter I
Introduction

Determining the optimal plan in constructing a telecommunication infrastructure or a transportation network could be regarded as a challenging task as it requires multi-dimensional management-competencies. This is known as the network design problem. The network design problem (NDP) is a problem of extracting the sub-network from a given network such that a budget constraint is satisfied and the total cost of the extracted sub-network is minimized comparing to other possible sub-networks [1]. The variations of the NDP were intensively reviewed by Magnanti and Wong [2].

The optimum communication spanning tree (OCST) problem is a particular case of the network design problem. It is one of the well-known combinatorial optimization problems widely studied by various optimization researchers across the globe. The problem was introduced by Hu [3] in 1974. The goal of the OCST problem is to find a spanning tree of the minimal total communication cost satisfying a given set of communication requirements. Since the last decade, the problem has increasingly gained more attention, as there is a broad range of applications in telecommunication, transportation, and computer network. The OCST problem is similar to the minimum spanning tree problem, but its constraint restriction made it more complex and its optimal solution cannot be obtained within polynomial time. In fact, it was proved to be NP-hard by Johnson et al. [4]. The application of the OCST problem was described in [5] as the constructing of the telecommunication line based on the tariffs of the German Telekom. In addition, identifying the OCST appears as a subroutine in one of the network hub location problems, called the Tree-of-Hubs Location Problem (THLP), which is a problem of locating a set of hubs and designing hub network such that the set of hubs are connected in the tree form [6]. The THLP can be applied in various areas such as the transshipment system, airline service and communication network.

Since 1974, there are three main research directions for dealing with the problem [7]. The first one is the approximation approach, which was proposed for various types of the OCST, e.g., Peleg and Reshef [8], Wu et al. [9], and Sharma [10]. The second approach is to use the optimization technique to solve for the exact solution of a small-size problem, for example, in [1], [11]. The last approach and the most prominent one since the year 2000 is to apply the heuristic method which was appeared in [12], [13] and [14]. Although most exact algorithms were based on the branch-and-bound approaches and MIP formulation worked well on moderate size networks, the algorithms is unsatisfactory for the larger networks [11]. Therefore, the heuristic approach for the OCST problem is broadly discussed by many researchers, as it was successful in obtaining the quality solution in a reasonable computational time.

Similar to other combinatorial optimization problems, the genetic algorithm, one of the metaheuristic approaches, has been intellectually applied in searching for the optimal solution of the OCST problem. The genetic algorithm (GA) was first proposed by Holland [15] in 1975 and has been claimed that the algorithm yields considerably good result in various types of problems, especially of combinatorial optimization problems [12]. The motivation behind the genetic algorithm comes from the principle of evolution and natural selection. The most critical step in executing the GA is how a candidate solution be represented. A solution in GA is usually transformed to the chromosome-like structure, as referred to the *encoding*. For the problem whose solution is in a form of tree structure, a number of tree representations must be used as in [12]. However, Palmer [12] also introduced a new representation called the *node and link biased encoding*. He claimed that his encoding is the best among others as it could prevent the tree-cycle formation and it does cover the desired search space of the solutions. The encoded solutions will be evolved by manipulating the “reproduction operators”; which includes the selection, crossover, and mutation. His work guided us to use an efficient way to represent a tree-structure solution.

According to the innovative idea from Palmer’s work [12], Hoang et al. [16] proposed a novel approach for the OCST problem, which was based on the particle swarm optimization-based (PSO) technique with node biased encoding (NBE) scheme. Their experimental results outperform the previous two results produced by

Palmer's GA [12] and Li and Bouchebaba's GA [17] by obtaining a comparative solution in a reasonable computational time. Improving their works to obtain the optimal solution or near optimal solution based on Hoang et al.'s approach would be aimed of this thesis.

In this work, we extend the concept of the particle swarm optimization-based algorithm of Hoang et al. [16] to the OCST problem by combining the concept of adaptive inertia weight strategy to velocity update step. In addition, our proposed algorithm is based on the fact that the best solution is biased toward a minimum spanning tree and initialized particles make a strong impact on the final solution [18]. We include a 0,1-vector to the initialized population whose components are filled by assigning 0 to an interior node and by assigning 1 to a leaf node. Executing our algorithm on a set of standard benchmark instances will exhibit the applicability of our approach to the OCST problem.

1.1 Objective

We aim to obtain an improved solution quality from Hoang et al.'s novel particle swarm optimization-based algorithm [16] for the optimum communication spanning tree (OCST) problem by combining the new adaptive inertia weight strategy proposed by Nickabadi et al. [19].

1.2 Thesis overview

In Chapter 2, we will discuss the literature survey related to the network design problem, the OCST problem including the problem definition, and the particle swarm optimization. In Chapter 3, the main algorithm will be explained. In Chapter 4, the experimental results will be presented and discussed. In Chapter 5, the conclusion will be stated.

Chapter II
Literature review

This chapter provides discussions among various publications related to the OCST problem, the particle swarm optimization, and the adaptive inertia weight strategies. The particle swarm optimization is explained mainly in OCST-applicable viewpoint. Various methods of solution representation used for solving tree optimization problem are also discussed.

2.1 Optimum communication spanning tree problem

The optimum communication spanning tree (OCST) problem definitions and details will be described as follows.

Let $G=(N,A)$ be an undirected weighted network, where N is a set of n nodes and A is a set of m arcs. Let a_q denotes the q^{th} arc of the network that can be written in the form $[i_q, j_q]$, when $q \in \{0,1,2,\ldots,m-1\}$. Let $d_{i_q j_q}$ denotes the length of a_q, or the distance between nodes i_q and j_q, which is represented in the matrix form $D=[d_{i_q j_q}]_{n\times n}$. We assume that the considering network is a complete graph; otherwise, the infinity value will be assigned to the arcs that do not exist in the network. Let r_{ij} denotes a communication requirement between nodes i and j, which can be interpreted as the multiplicative path between a pair of nodes that are needed in the real communication network. A set of communication requirement is represented in the matrix form $R=[r_{ij}]_{n\times n}$, where $i \in \{0,1,2,\ldots,n-1\}$ and $j \in \{0,1,2,\ldots,n-1\}$.

The communication cost $c_{ij}(T)$ for a pair of nodes i and j over a spanning tree T is calculated as the product of the communication requirement to the length (shortest distance) of the unique path between two nodes in T:

$$c_{ij}(T) = r_{ij} \cdot \sum_{[p,q]\in Path(i,j)} d_{pq}$$

where $Path(i,j) = \{[v_z, v_{z+1}] \in T \mid v_1 = i, v_{\hat{z}+1} = j, \text{ and } z = 1,2,\ldots,\hat{z}\}$ denotes the unique path between the nodes i and j over the spanning tree T, and $\hat{z}$ denotes the number of arcs on the unique path between the nodes i and j.

Consider the Palmer6 network instance, which is a network consisting of six nodes with a distance matrix

$$D = \begin{bmatrix} 0 & 16661 & 18083 & 21561 & 21099 & 13461 \\ 16661 & 0 & 5658 & 9194 & 8797 & 10440 \\ 18083 & 5658 & 0 & 7230 & 6899 & 11340 \\ 21561 & 9194 & 7230 & 0 & 4300 & 13730 \\ 21099 & 8797 & 6899 & 4300 & 0 & 13130 \\ 13461 & 10440 & 11340 & 13730 & 13130 & 0 \end{bmatrix}$$

and a communication requirement matrix $R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 0 & 10 & 3 & 4 & 3 \\ 1 & 10 & 0 & 5 & 6 & 2 \\ 1 & 3 & 5 & 0 & 31 & 2 \\ 1 & 4 & 6 & 31 & 0 & 2 \\ 2 & 3 & 2 & 2 & 2 & 0 \end{bmatrix}$. The network is shown in Figure 2-1 below.

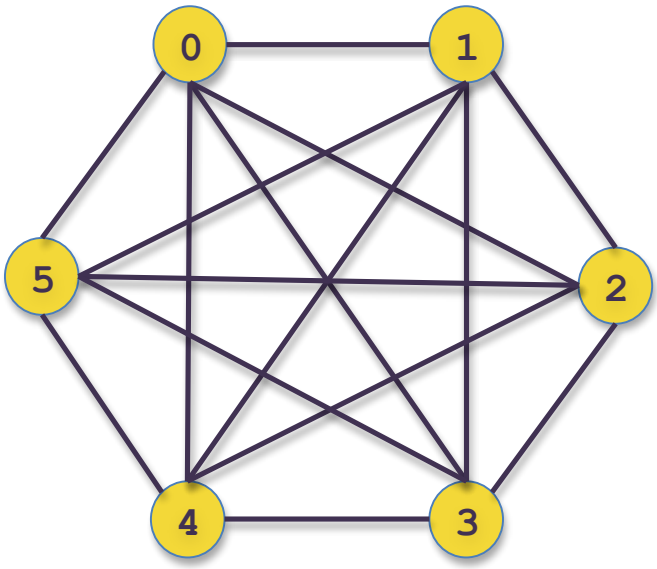

Figure 2-1: Diagram represents the Palmer6 network

For example, the distance between node 0 and node 1 is 16661 and the distance between node 2 and node 4 is 6899. As it is treated as an undirected weighted graph, the matrix D is symmetric, and the matrix R is also symmetric. For instance, the communication requirement between node 0 and node 1 is 1, and the communication requirement between node 2 and node 4 is 6.

A spanning tree T, for example, extracted from the Palmer6 network, is shown in Figure 2-2. The communication cost between node 1 and node 4 is $c_{14}(T) = r_{14} \cdot (d_{13} + d_{34}) = 4 \cdot (9194 + 4300) = 53976$.

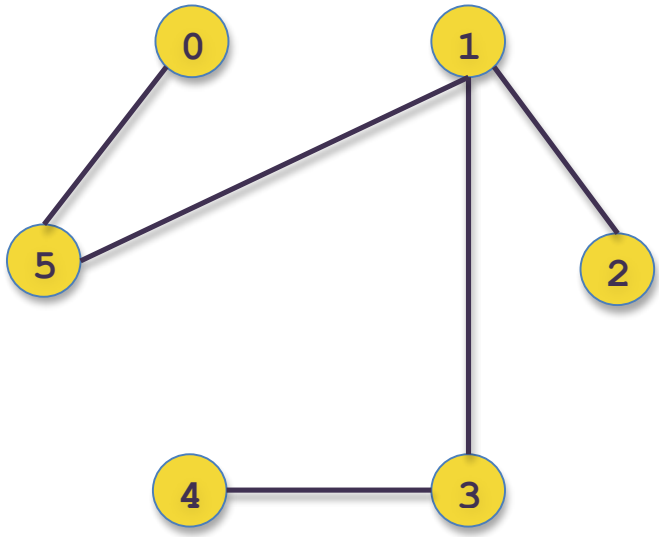

Figure 2-2: Diagram represents a spanning tree T extracted from the Palmer6 network

Therefore, the total communication cost $C(T)$ of a spanning tree T is obtained by summing the communication cost over all pairs of nodes:

$$C(T) = \sum_{\substack{i<j \\ i,j \in Q}} c_{ij}(T)$$

where Q denotes the set of all pairs of nodes of the spanning tree T.

The goal of the OCST problem is to construct a spanning tree connecting all nodes in N such that the total communication cost of the spanning tree is minimum among all spanning trees of G.

In 1974, Hu [3] introduced a new type of spanning tree problems, which was considered not only the distance d_{ij} between two nodes but also the requirement r_{ij} of communication between a pair of nodes. However, Hu inspected only two specific cases; when all distances were set to one unit, which was named as the "optimum requirement spanning tree", and when all requirements were set to one unit, which was named as the "optimum distance spanning tree". The concept of max-flow min-cut theorem was employed in solving the first case while the motivation of constructing the star-tree was used for the latter case.

In 1987, the exact and heuristic algorithms for the OCST problem were proposed by Ahuja and Murty [1]. For the exact algorithm, the branch-and-bound method was used along with the new lower planes. To obtain the lower bound for the branch-and-bound algorithm, the algorithm needs all weights for the lower plane, which can be computed in $O(n^4)$ time. Accordingly, the computation is unacceptable when dealing with a large network. For this reason, they also developed a heuristic algorithm, which consists of two phases: the tree-building phase and the tree-improvement phase. The tree-building phase was based on the greedy approach, which the tree arcs were spanned through the least communication cost arcs, while the tree-improvement phase was based on a local search method from the one tree-arc exchange routine until all arcs were once examined. The two-phase heuristic algorithm required $O(n^3)$, so it would be more suitable for a large network. Besides, they also presented a new heuristic algorithm called the "global heuristic algorithm", which was mainly based on the branch-and-bound algorithm with strategy for modifying the lower bound value. They concluded that the lower bounding strategy has a massive impact and the two-phase heuristic algorithm gives the excellent result.

In 2006, Sharma introduced a pseudo-polynomial algorithm by constructing a near optimum tree with the concept of cut-tree from Hu's method [10]. The near optimum tree concept is to consider the neighborhood of a tree that differs from it in only one arc. The tree with minimum total communication cost among the neighborhood is recorded as the near optimum tree. The pseudo-polynomial algorithm is initialized by setting all arc distances to be equal to the smallest distance of the considering network and the starting tree can be constructed by Hu's max-flow min-

cut method for the “Optimum requirement spanning tree problem”. For an arc which is not in the current tree, the associated arc distance is undoubtedly increased to its actual distance value as it will not affect the total cost of the tree. For a tree arc with distance value which is less than the actual distance value, the current tree is updated by exchanging (adding and then deleting) one arc that yields the minimum cost and the associated arc distance is increased by computed delta. The algorithm would be repeated until all arc distances reach their actual distance values. Moreover, when the distances of the network satisfy the generalized triangular inequalities, another algorithm which called *OCSTP II*, was used in constructing the near optimum tree. The *OCSTP II* was similar to the first algorithm but, instead of d_{ij}, all r_{ij} were set to be the smallest r_{ij} at the beginning and then increased one at a time until all arcs attain their actual r_{ij} values. Although this paper does not show the running result on the benchmark instances, the paper verifies that the pseudo-polynomial algorithm requires $O(M \cdot n^4)$, where M is the maximum arc distance of the network, and the *OCSTP II* algorithm needs $O(n^3 \cdot m)$.

One of the first evolutionary algorithms for solving the OCST problem was performed by Palmer [12] in 1994. Palmer introduced the genetic algorithm (GA) to solve the OCST problem. He identified the appropriate encoding criteria in representing the tree which would give the best result for the GA. Although various tested encodings produced unimpressive results, the most encourage one, called *node and link biased encoding*, had been experimented by assigning a weight, which is called the *bias* value, to the node and/or the arc. Such encoding follows by the intuition that certain node could be an interior node or a leaf node. However, including the arc-biases in the encoding required the long schemata on a chromosome and the sense of being whether an interior or leaf node could be reflected from the node-bias value. So, Palmer ignored the effect of the arc-biases by setting the arc-bias control parameter to zero. In other words, by his comment, the most appropriate encoding for the GA was actually the *node biased encoding*. Their experiment with such encoding provided better results than those obtained by any standard heuristic.

In 1999, Li and Bouchebaba [17] proposed the new genetic algorithm for the OCST problem. Their algorithm had two enhancements from the Palmer's GA. First, Li and Bouchebaba's algorithm did not deal with chromosome encoding-decoding on nodes and/or links like Palmer's algorithm did. A tree structure was used as a chromosome of their GA to improve the searching ability. Second, the initial population were created randomly based on Prim's algorithm. They claimed that doing so would result to the near-optimum initialized solution, which will definitely be easier to reach the optimal solution by employing the crossovers and mutations as shown in their computational results. However, their proposed idea produces considerably large gap between the obtained solution and the best known solution in most benchmark test instances.

In 2007, Fischer and Merz introduced the new evolutionary algorithm (EA), which used the local search within the EA [14], called *Memetic Algorithm* (MA). However, the discussion of this work was mainly on the evaluation of various recombination operators. In addition, various concepts of constructing the initial population have been examined with both the Raidl-series instances and random instances. The results show the advantage of their approach over previous works.

In 2010, Hoang et al. [16] proposed a PSO-based algorithm for the OCST problem. They notified the four main components of the algorithm. First, the most important component was the spanning tree representation, which they employed the node-biased encoding. Another component was the population initialization. The algorithm started with a particle whose structure is a minimum spanning tree, other n particles which corresponds to the star-shaped tree having one of the nodes as an interior node, and other $psize-(n+1)$ particles whose components were randomly generated from real number between 0 and 1, where $psize$ is a population size in the algorithm. Another key component of PSO was the fitness function, which was calculated as the proportion of the total communication cost of a tree. The termination condition was required to stop the algorithm. By running on a number of OCST benchmark instances, the algorithm produced improved results on most tested instances. Nonetheless, the obtained solutions for instances with more than twenty nodes were not impressive, as there were large gaps between the obtained solutions and their corresponding best known solutions.

In 2011, Rothlauf [18] inspected the bias of the OCST benchmark test instances. The study was based on the believe that if the elements of the heuristic algorithm, e.g., the solution representation, the fitness function, the initial solutions, and the search strategy, were designed by satisfying the specific characteristic of individual problem, such heuristic algorithm can produce the high quality solution and is titled as the *biased modern heuristics*. Rothlauf also reviewed various types of solution representation for the OCST problem. That publication showed that the best solution of the OCST problem was similar to MST and well-initiated solution could lead to a near optimal solution. This is a motivation of introducing a new initial particle generated by the minimum spanning tree to our proposed algorithm.

2.2 Particle swarm optimization

Particle swarm optimization (PSO) algorithm is a population-based heuristic optimization technique, which was originally presented by Kennedy and Eberhart [20] in 1995. It is inspired by the concept of intelligent collective behavior of bird flocking and fish schooling. The algorithm shares many commonalities with other evolutionary algorithms such as Genetic Algorithm (GA). However, only the global version of the PSO, such that particles moved toward position of the “particle best” position and “global best” position in the hyperspace, had been introduced. Consequently, the same authors also developed the local version such that, in addition to the “particle best”, the position of the best solution among the nearest neighbor in the hyperspace, called the “local best”, was considered instead [21]. By comparing to other evolutionary-based optimizations, the PSO is easy to implement and has few parameters to adjust. The key factors of the PSO algorithm in OCST problem are explained below [16].

2.2.1 Spanning tree representation

As mentioned earlier, representation of a solution is one of the major factors that have a strong impact to the final result [12]. An effective encoding should possess the following properties

1. Representing all possible spanning trees
2. Unbiased
3. Capable of representing only trees
4. Easy to go back and forth between the encoded representation and the tree representation
5. Employed with short, low order schemata
6. Possess locality

There are various methods to represent the solution in the tree-structure, e.g., the characteristic vector encoding, the predecessor encoding, and the Prüfer number [22]. There are different advantages and disadvantages among different representations.

For the characteristic vector encoding, the vector of the same size as the number of arcs is used. The most expensive case occurs when the considering graph is a complete graph by utilizing the vector of size $\frac{n \cdot (n-1)}{2}$, where n is the number of arcs in the graph. Each component is filled by 1 if the corresponding arc is included in the tree, otherwise, it is filled by 0. Since the desired solution is a spanning tree having only $n-1$ arcs, this encoding could be easily produced non-tree solution when a number of 1s is more than $n-1$, which guarantee that at least one cycle is formed. This encoding performs well only on the small network.

For the predecessor encoding, the predecessor of each node is recorded. The probability to form the tree by this encoding is higher than using the characteristic vector encoding. However, this encoding can still easily produces the non-tree solution both in the initial population and during the evolution process.

The Prüfer number, $P(T)$, uses string of size $n-2$ to represent a spanning tree, starting with the empty string and with the original tree, by recognizing the predecessor of the lowest numbered leaf node of the current tree and then appending the predecessor node number to the right-most digit of the constructing $P(T)$. When the $P(T)$ is updated, the lowest numbered leaf node and the arc connecting the node and its predecessor are removed from further consideration. The process is repeated until only two nodes remain for consideration, then the Prüfer number has been found.

The encoding was developed as a one-to-one mapping between a spanning tree solution and the Prüfer number. This encoding is produced only a tree and is unbiased. However, this encoding possess low locality, as the offspring formed by two good parents can be extremely different from the previous generation.

In 2011, Rothlauf [18] reviewed various types of solution representation for the OCST problem, including the category of direct representation which uses the non-encoding tree structure in implementation, e.g., the *edge-set encoding* and the *NetDir encoding*.

The edge-set encoding uses a set of arcs to represent a tree [23]. Due to the importance of population initialization in the heuristic algorithm, an algorithm for producing random spanning trees, which can be considered as a function of the recombination process among population, is required. Three different techniques, i.e., PrimRST, KruskalRST, and RandWalkRST, were tested. The PrimRST is based on Prim's algorithm for MST, but, instead of appending new arc with criteria of the least cost arc, randomly choosing new appending arc which is adjacent to a node in the constructing tree. Although the PrimRST is an easy way of constructing random spanning trees, there is more chance to produce star trees. Moreover, the trees from PrimRST are biased to some tree structure and some path structure such that are not uniformly distributed. The KruskalRST is based on Kruskal's algorithm for MST except that an appending arc is chosen in random order. The same disadvantage as in PrimRST occurs such that there is higher chance to produce some particular tree structures than others. To avoid the bias behavior in initialization, the new method based on random walk, called RandWalkRST, was brought to the process. By starting at an arbitrary node in the graph, the tree is appended by randomly choosing adjacent arc from the current node. When the new node is visited for the first time, such arc will be connected to the constructing tree. The appending process is repeated until all nodes of the considering network has been visited. The NetDir encoding is a method of representing trees by deploying the algorithm directly to the native graph. However, problem-specific operators are needed.

Although performance of utilizing the direct representation is satisfactory, discovering appropriate search operators that make the search space uniformly spread is intricate.

For the OCST problem, Palmer [12] proposed the node and linked biased encoding scheme, one of the classes of weighted linear chromosome representation for spanning tree optimization problems. However, the node-only version of the encoding was employed because it requires the string of size as short as the number of nodes of the considered network, which is much less than the number of arcs, while retain the basic concept behind this encoding. The node-biased encoding was described as follows [16].

Each spanning tree is represented by a vector of size n, where n is a number of nodes in the network. Each component of a vector, which has its own label correspond to one of n nodes of the network G, is a real number between 0 and 1. The details of encoding and decoding are given below.

Encode phase: Each spanning tree T of the network $G=(N,A)$ is represented by n-component real vector $b=(b_0,b_1,\ldots,b_{n-1})$, called *weighted vector*, where n is the total number of nodes in G and $b_i \in [0,1]$.

Decode phase: Calculate a spanning tree corresponding to the weighted vector b by the following steps

Step 1 Construct the associated network $G'=(N,A)$ with modified distance matrix $D'=[d'_{i_q j_q}]_{n\times n}$. It is computed from the following equation

$$d'_{i_q j_q} = d_{i_q j_q} - p \cdot d_{\max} \cdot (b_{i_q} + b_{j_q}) \qquad (2.1)$$

where $d_{\max}$ is the maximum distance of any arcs of G, and p is the node biased parameter which controls influence of the biases to the constructed tree.

Step 2 Calculate a spanning tree T as the minimum spanning tree of the associated network G' using Prim's algorithm [24].

For example, we demonstrate the weighted vector $b=(1,0,1,0.78,1,0.69)$ of the Palmer6 network. Each weight value is assigned to the associated node as in the Figure 2-3. The modified distances of the modified graph G' are calculated following the equations (2.1) and this step is represented in Figure 2-4. Then, the Prim's algorithm is used for extracting a minimum spanning tree from the G'.

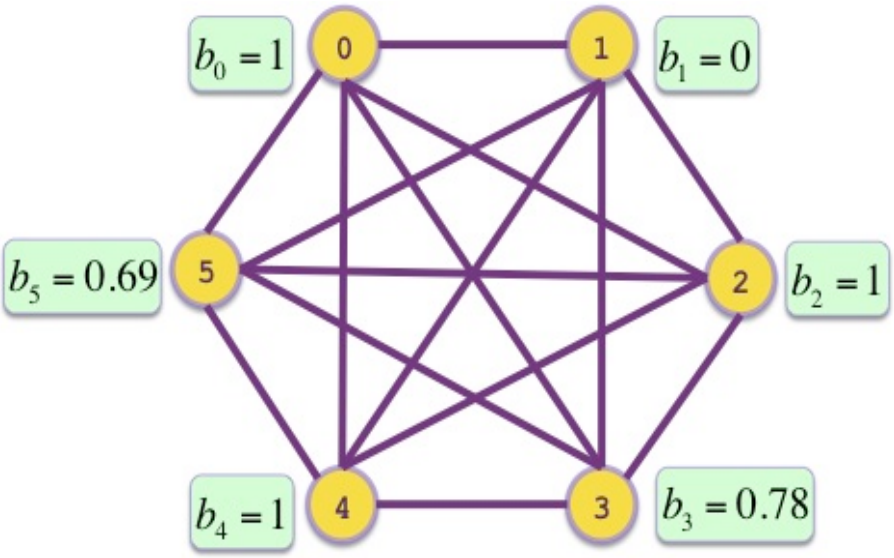

Figure 2-3: Diagram represents the Palmer6 network with weights

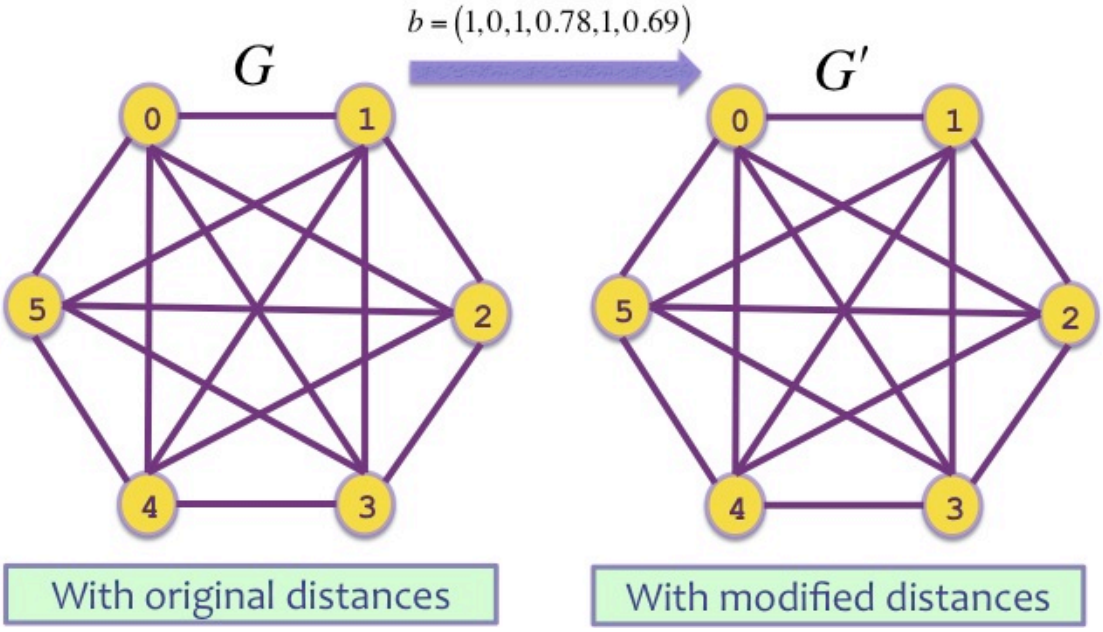

Figure 2-4: Diagram shows *Step 1* of the decode phase for the Palmer6 network

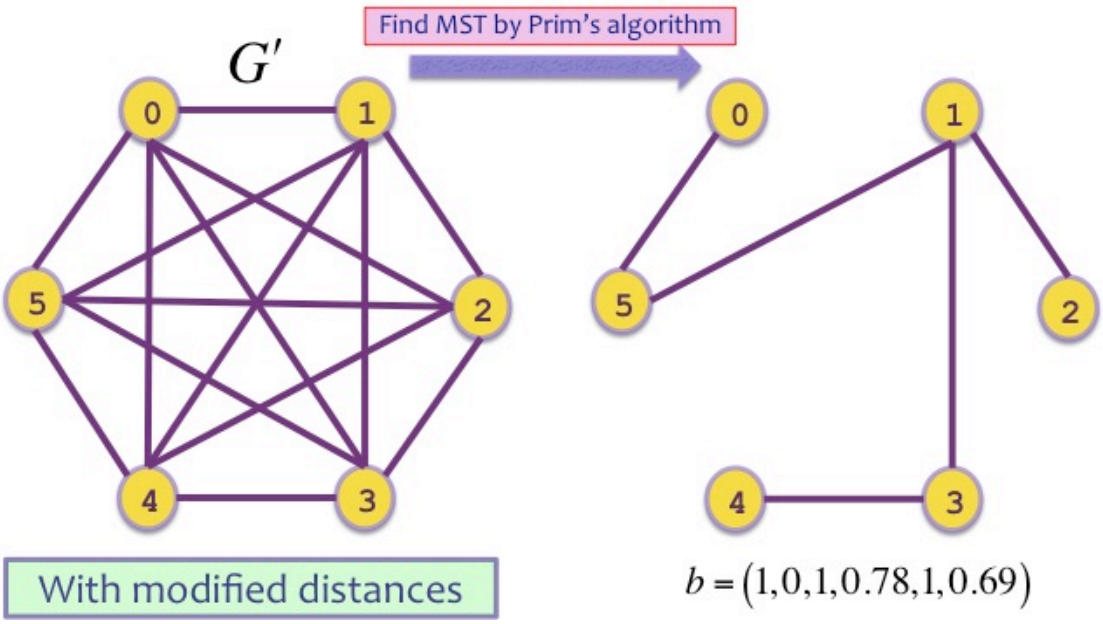

Figure 2-5: Diagram shows *Step 2* of the decode phase for the Palmer6 network

To apply the PSO algorithm to the OCST problem, a set of candidate solutions has to be initiated, which will be explained in the next subsection. Each of a single solution is a "bird", as in the PSO algorithm called "particle". Each particle is treated as a point, represented by a vector "position", in an n-dimensional space, where n is the number of nodes in the network. A "swarm", a set of particles in a current state which is also known as a "population", flies through the n-dimensional space in which the position of each particle is improved according to its own experience and the experience of its neighbors. To move a particle from one position to another, the algorithm requires "velocity" to indicate how the particles should fly through the search space.

By following Palmer's node-biased encoding described above, the position of the k^{th} particle (in the n-dimensional space) is represented as $x_k = (x_{k0}, x_{k1}, \ldots, x_{k(n-1)})$, where $x_{kd} \in [l_d, u_d], d \in [0, n-1]$ and l_d and u_d are the lower and upper bounds of the particle in the n-dimensional space. The velocity of the k^{th} particle is represented as $v_k = (v_{k0}, v_{k1}, \ldots, v_{k(n-1)})$. Let $newx[k]$ denotes the new position of the k^{th} particle and $currentx[k]$ denotes the current position of the k^{th} particle. Let $newv[k]$ denotes the

new velocity of the k^{th} particle and *currentv*[k] denotes the current velocity of the k^{th} particle.

Each particle with any specific position posses a *fitness value*, which is evaluated by the *fitness function*. In each iteration, each particle is updated by two best values, i.e., *pbest* and *lbest*. The *pbest* value is the best value of the fitness function that has been achieved so far by any particle, while the *lbest* value is the best fitness value of all neighbor particles. The number of neighborhood is given by a user. After finding the two best values, the k^{th} particle updates its velocity and position by the following equation

$$\begin{aligned} newv[k] = currentv[k] + c_1 \cdot R_1 \cdot (pbest[k] - currentx[k]) \\ + c_2 \cdot R_2 \cdot (lbest[k] - currentx[k]) \end{aligned} \tag{2.2}$$

$$newx[k] = currentx[k] + newv[k] \tag{2.3}$$

where R_1 and R_2 are two distinct random numbers in [0,1], and c_1 and c_2 are acceleration constants.

2.2.2 Population initialization

In Hoang et al.'s work, they initialized the first particle with one zero-vector, which correspond to a MST according to non-modified distance matrix. Then, another n particles were motivated by the star-shaped tree having value 0 in only one component, otherwise having value 1. All other $psize-(n+1)$ particles were vectors whose components were randomly generated real number from the interval [0,1], where *psize* is a population size in the algorithm. Therefore, to maintain the full concept of this initialization pattern, the population size should set larger than $(n+1)$. In other words, $psize-(n+1)$ must greater than zero.

2.2.3 Fitness function

Let C_k be the total communication cost of the k^{th} particle.

Let F_k be the fitness value of the k^{th} particle which was computed as

$$F_k = \frac{1}{C_k}.$$

2.2.4 Termination condition

The termination condition was defined as the maximum number of iterations.

The overview of the PSO algorithm is shown in Figure 2-6 below.

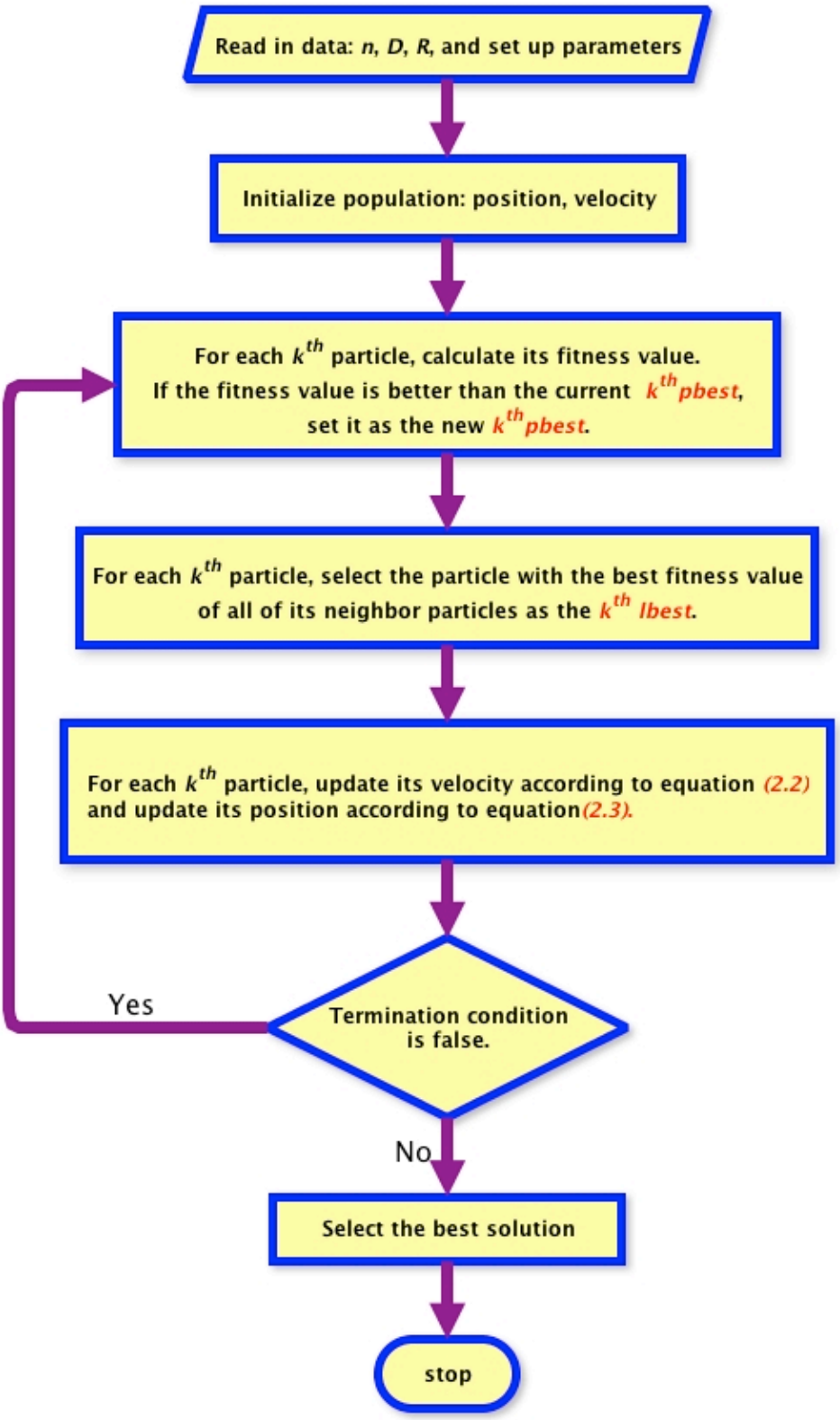

Figure 2-6: Flowchart shows the PSO algorithm

2.3 Particle swarm optimization with adaptive inertia weight

Although the process of the original PSO algorithm gives particles appropriate direction toward the optimal solution, it can be stuck on a local optimum in the case when the first term of the velocity update equation, i.e., the *currentv*[k] term in the equation (2.2), is near zero [25]. When the first term of the velocity update equation is omitted, the flying space will be contracted to the current best solution which will lead to the unpleasant result of a local optimum. The optimal solution can be reached only when the optimal solution is one of the initial solutions. So, the principal role of the *currentv*[k] term is to enhance the global search ability, which could lead to the global optimum, whereas the role of the second and the third terms are to determine the direction heading to the best position in history, which can be considered as a local search.

Shi and Eberhart [25] state "the balance between global and local search throughout the course of a run is critical to the success of an optimization algorithm." Accordingly, they brought the inertia weight into the PSO algorithm in 1998 [25]. The inertia weight (ω) is one of PSO's parameters as global search and local search balancing mechanism and is included in the velocity update equation as follows.

Let k^{th} be a particle, the velocity update equation is

$$\begin{aligned} newv[k] = \omega \cdot currentv[k] + c_1 \cdot R_1 \cdot (pbest[k] - currentx[k]) \\ + c_2 \cdot R_2 \cdot (lbest[k] - currentx[k]) \end{aligned} \tag{2.4}$$

where R_1 and R_2 are two distinct random numbers in [0,1], and c_1 and c_2 are acceleration constants and ω is an inertia weight.

In their experiment, they tested on the two dimensional Schaffer's function with different inertia weight settings. The algorithm was run 30 times for each fixed inertia weight. The number of iterations required in obtaining the optimal solution and the number of runs that could not find the optimal solution within 4,000 iterations were recorded. They found that the best range of inertia weight for this experiment is in the range of $(0.8,1.2)$. However, the best performance in the experiment was obtained by the time varying inertia weight, i.e., the linear decreasing function over iterations. This could produce the best known solution in every single run and average

number of iterations required until reaching the best known solution was lower than those from the fixed constant inertia weights.

Different strategies for determining the value of inertia weight during the course of run have been widely discussed by many researchers. Using an appropriate strategy of the inertia weight would improve algorithm performance in obtaining the accurate solution. In 2011, Nickabadi et al. [19] reviewed various published inertia weight strategies and proposed a new adaptive inertia weight strategy, which uses the success rate of the swarm as its feedback parameter to determine the state of the particles in the search space at each run. In terms of convergent speed and solution accuracy, their new inertia weight strategy performed better on some test functions than other previous works. In their review, the inertia weight strategies could be classified into three main classes.

1. Constant and random inertia weights

In this class of the inertia weight, the inertia weight is fixed or is randomly generated in a form of functions over all iterations. The advantage of the constant inertia weight strategy is that it is easy to include into the velocity update equations; however, this strategy is not suitable in a dynamic environment. The random inertia weight is used for improving the search in a dynamic environment.

2. Time varying inertia weights strategies

The time varying inertia weight strategy has been used in most PSO algorithm. The most notable one of this strategy is the linear decreasing inertia weight strategy. In this strategy, the value of inertia weight is derived by the linear decreasing function over iterations. The value of inertia weight is linearly decreasing from an initial value to a final value.

3. Adaptive inertia weights

In this strategy, the inertia weight is updated according to the feedback parameter on each iteration. Various feedback parameter has been proposed over the decades; however, Nickabadi et al. proposed the new adaptive inertia weight, which uses the percentage of the success of the swarm as feedback parameter. By running PSO algorithms with various inertia weight strategies on a number of benchmark functions, Nickabadi et al.'s new adaptive inertia weight produces improved solutions

comparing to previous inertia weight strategies in terms of both convergence speed and solution accuracy [19].

The success of the k^{th} particle at iteration t is defined as

$$S(k,t)=\begin{cases} 1 & \text{if } fitness(pbest[k]_t) > fitness(pbest[k]_{t-1}) \\ 0 & \text{if } fitness(pbest[k]_t) \le fitness(pbest[k]_{t-1}) \end{cases}$$

Therefore, the percentage of success of the swarm at iteration t is defined as

$$P_S(t)=\frac{\sum_{k=0}^{psize-1} S(k,t)}{psize}$$

The linear function is used for determining the inertia weight of iteration t $\omega(t)=(\omega_{\max}-\omega_{\min})P_S(t)+\omega_{\min}$.

As far as we know the OCST problem, the PSO algorithm, and the idea of the recently proposed adaptive inertia weight strategy, combining the new adaptive inertia weight into the velocity update equations would yield improved results. In the next chapter, the main algorithm of our proposed *AIW-PSO algorithm* for the OCST problem will be explained.

Chapter III
Adaptive inertia weight particle swarm algorithm for the OCST problem

Our proposed algorithm, which is called *AIW-PSO algorithm*, is based on Hoang et al.'s PSO algorithm as it was confirmed to be superior to two previous algorithms: Palmer's GA and Li and Bouchebaba's GA. Representation of the solution, calculation of fitness function, and termination condition is adopted from Hoang et al.'s work. The primary components, i.e., population initialization, velocity initialization, and measurement of dissimilarity between a pair of trees, of our algorithm will be explained below.

3.1 Population initialization

We use similar population initialization patterns by Hoang et al., except only one random particle will be replaced with the new 0,1-vector. As summarized by Rothlauf [18] that the best solution for the OCST is biased toward minimum spanning tree (MST) and good initial particle is a MST which can steer toward the optimal solution.

We adopt one initial particle which have a crucial effect on the final solution accuracy. We substitute one random vector with the 0,1-vector derived from the solved MST from original distances. We extract the degree of each node on the MST. If degree of node v is greater than 1, then the corresponding component will be set to 0, otherwise it is set to 1. The motivation behind this 0,1-vector is, with the node-biased encoding, the low weighted node has a tendency to be an interior node while the high weighted node tends to be a leaf node.

3.2 Velocity initialization

As far as we concern, the velocity at the initial iteration should not be random because the initial random velocity would destroy the randomness of the well-planned patterns of the initial particles, while starting with zero velocity will immediately steer the particles toward the best fitness particles at the first iteration. In addition, the study of velocity initialization by Engelbrecht [26] in 2012 concludes that "initial velocities should be set to zero, or small random values close to zero." Accordingly, we set the initial velocity as zero vectors.

3.3 The measure of dissimilarity between a pair of trees

Before we obtain the local best position of k^{th} particle for updating the velocity, we need to know the distance between each pair of solutions, which is indeed a pair of trees, in the search space. As the definition of the measure of dissimilarity was not mentioned in Hoang et al.'s work and we cannot use ordinary arithmetic with the tree structure solutions, so we make a new, but intuitive, way of measuring such dissimilarities. Because the solution of the OCST problem is actually a spanning tree, each tree structure solution has remembered its own associated arcs, which consists of $n-1$ arcs, when the number of nodes in the considered network is n.

The dissimilarity between any two trees is defined as a positive difference of the sum of the distances of their associated arcs of the two trees. The definition means, when a tree arc exists on both trees, such arc will not affect to the dissimilarity measure and this arc distance will not be included in dissimilarity value, otherwise the arcs will be taken into the calculation.

Let T_{k_1} and T_{k_2} be the tree solutions of k_1^{th} particle and k_2^{th} particle, respectively. We all know that the T_{k_1} consists of $n-1$ arcs, says $a_{k_1 0}, a_{k_1 1}, \ldots, a_{k_1 (n-2)}$, and the T_{k_2} consists of $n-1$ arcs, says $a_{k_2 0}, a_{k_2 1}, \ldots, a_{k_2 (n-2)}$. The dissimilarity between T_{k_1} and T_{k_2} is $\left| \sum_{h=0}^{n-2} d_{a_{k_1 h}} - \sum_{h=0}^{n-2} d_{a_{k_2 h}} \right|$, where $d_{a_{k_i h}}$ is the original distance of associated arc of T_{k_i}.

For example, consider two trees, T_{k_1} and T_{k_2}, from Palmer6. They are graphically represented in Figure 3-1 and Figure 3-2, respectively.

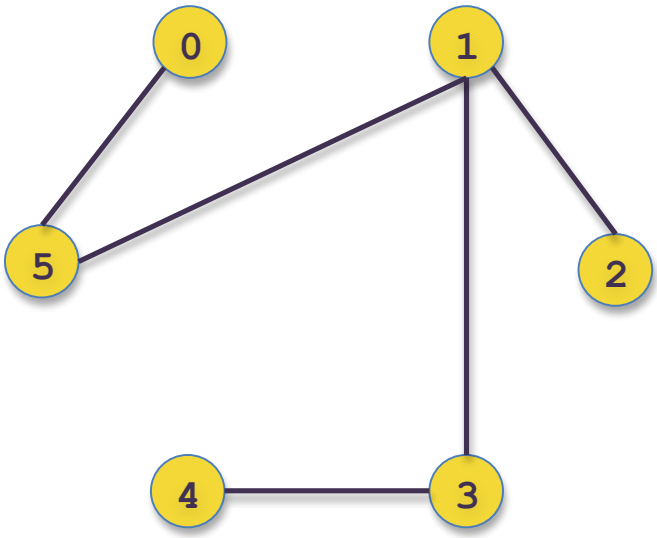

Figure 3-1: T_{k_1}, a tree extracted from the Palmer6 network

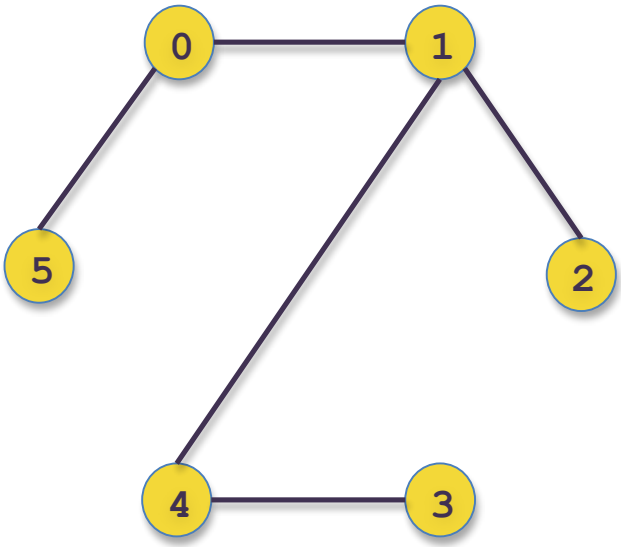

Figure 3-2: T_{k_2}, a tree extracted from the Palmer6 network

Refer to the distance matrix D showed in Section 2.1, the dissimilarity between T_{k_1} and T_{k_2} is $\left|(d_{05}+d_{12}+d_{13}+d_{15}+d_{34})-(d_{01}+d_{05}+d_{12}+d_{14}+d_{34})\right|$.

3.4 Adaptive inertia weight

We include the new adaptive inertia weight proposed by Nickabadi et al. The new adaptive inertia weight strategy uses percentage of the success as feedback parameter to determine the state of the particles in the search space at each iteration. The percentage of the success of the swarm is already described in Section 2.3.

3.5 Flowchart

To see the overview of the AIW-PSO algorithm for the OCST problem, we represent the algorithm as a flowchart in Figure 3-3.

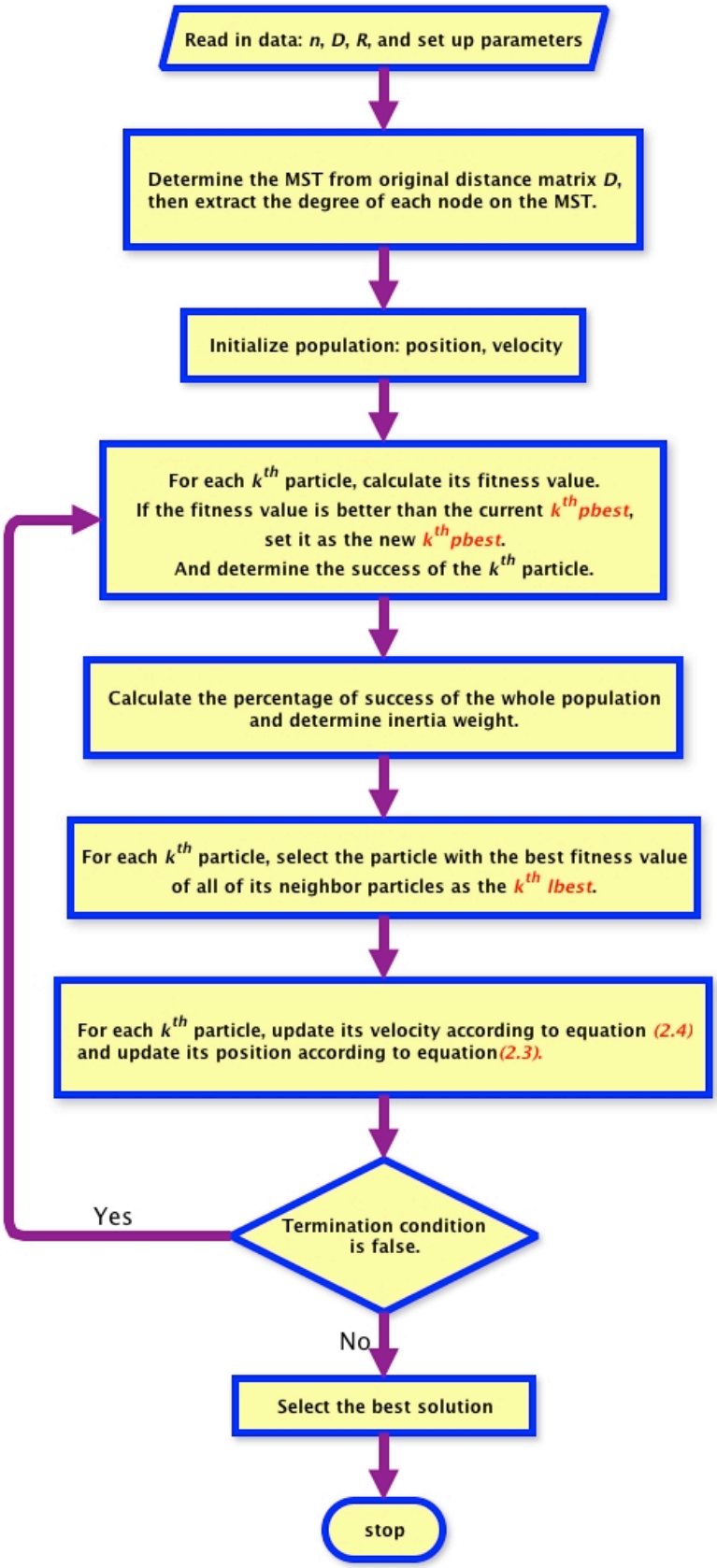

Figure 3-3: Flowchart shows our AIW-PSO algorithm for the OCST problem

Chapter IV
Experimental results

There are a number of OCST benchmark instances that had been used for evaluating the algorithm efficiency. We test our AIW-PSO algorithm on ten OCST benchmark instances. These are 6 nodes, 12 nodes, and 24 nodes networks from Palmer; 10 nodes, 20 nodes, 50 nodes and 75 nodes networks from Raidl; and 6 nodes, 35 nodes (Berry35, Berry35u) networks from Berry. These instances can be found in Rothlauf [5].

The Palmer's instances were created based on the actual data of the U.S.'s inter-city transportation. The inter-city link cost was obtained from the tariff database, while the inter-city requirements were generated based on a number of information such as the population in each city and the distances between the cities. Another set of OCST benchmark instances is from Raidl. The Raidl's distance costs and communication requirements were randomly generated with uniform distribution. The third set of OCST benchmark instances is from Berry et al. All link costs in the Berry35u were set to one, but the communication requirements were the same as the Berry35. A number of parameters are determined as follows.

Designed parameters

Population size $psize$ = 1000.

Maximum number of iteration $N_{\max}$ = 500.

Node biased parameter p = 1.

From the velocity update equation (2.2), cognitive and social parameters c_1 = 1 and c_2 = 1.

Number of neighbors for local best (lbest) K = 3.

Our AIW-PSO algorithm is coded in C++ and run on 1.86 GHz Intel Core 2 Duo with 4GB RAM. Our initialization of the population size *psize* consumes $O(psize \cdot n)$ as each particle requires n random values, also the same complexity for the velocity update and position update. For the spanning tree decoding procedure, step 1 obviously requires $O(n^2)$ as we have to compute the modified distances between every possible pairs of nodes. Step 2 of the decoding procedure consumes $O(n \cdot \log m)$ as we implement Prim's algorithm by using min heap structure. For the measure of dissimilarity between two trees, this part needs $O(psize^2 \cdot n)$ as we need to calculate such dissimilarities between every pairs of trees from the whole population, and, for a pair of trees, all $(n-1)$ arcs for each tree is considered for the calculation. To determine the local best for a particle, those dissimilarity values for the particle are sorted by calling function *qsort*() from the standard library which needs $O(psize \cdot \log(psize))$ for the average case. So, determining local best for the whole population requires $O(psize^2 \cdot \log(psize))$. Therefore, the total complexity of our algorithm with population size *psize* and maximum number of iterations $N_{\max}$ is $O(N_{\max} \cdot psize \cdot (n^2 + n \cdot \log m + psize^2 \cdot n + psize^2 \cdot \log(psize) + psize \cdot n)) =$ $O(\max\{psize \cdot n^2, psize^3 \cdot n\})$.

The algorithm is tested on each instance for five times. The most often occurred best solution of five runs was recorded as an "obtained solution" for each instance. However, some previous algorithms were not tested on all available OCST benchmark instances; for example, Hoang's PSO was not tested on Berry35 and Berry35u. So, "n/a" will be reported. The numerical results are presented in Table 1.

Table 1 shows that our algorithm is as good as Hoang's algorithm as both algorithms produce best known solutions for Palmer6, Palmer12, Raidl10, Raidl20, and Berry6. In addition, our algorithm yields the obtained solutions which being closer to the best known solutions for Palmer24 and Raidl50 and produces best known solutions for the Berry35 and Berry35u, which was not mentioned in Hoang et al.'s paper.

Table 1: The obtained solution (Total Communication Cost) for the OCST benchmark instances

Instances	Best known solution	Palmer's GA	Li & Bouchebaba's GA	Hoang's PSO	AIW-PSO
Palmer6	693,180	709,770	708,090	693,180	693,180*
Palmer12	3,428,509	3,876,488	3,457,952	3,428,509	3,428,509*
Palmer24	1,086,656	1,959,790	1,086,656	1,138,360	1,088,154
Raidl10	53,674	58,352	53,674	53,674	53,674*
Raidl20	157,570	165,788	157,570	157,570	157,570*
Raidl50	806,864	911,987	964,140	826,499	809,311
Raidl75	1,717,491	n/a	n/a	n/a	1,730,153
Berry6	534	534	534	534	534*
Berry35	16,915	n/a	16,915	n/a	16,915*
Berry35u	16,273	n/a	16,420	n/a	16,273*

(* denotes the best known solution for each instance)

Table 2: The percentage of gap between the obtained solution and the best known solution

Instances	Palmer's GA	Li & Bouchebaba's GA	Hoang's PSO	AIW-PSO
Palmer6	2.3933%	2.1510%	0.0000%	0.0000%
Palmer12	13.0663%	0.8588%	0.0000%	0.0000%
Palmer24	80.3505%	0.0000%	4.7581%	0.1379%
Raidl10	8.7156%	0.0000%	0.0000%	0.0000%
Raidl20	5.2155%	0.0000%	0.0000%	0.0000%
Raidl50	13.0286%	19.4923%	2.4335%	0.3033%
Raidl75	n/a	n/a	n/a	0.7372%
Berry6	0.0000%	0.0000%	0.0000%	0.0000%
Berry35	n/a	0.0000%	n/a	0.0000%
Berry35u	n/a	0.9033%	n/a	0.0000%

To assure the effectiveness of our algorithm, we present our results by the percent improvement. The percent improvement of an algorithm on an instance is measured by the percentage of the gap between the obtained solution and the best known solution of an instance. The best known solutions can be found in [18]. Such measurement means that the less of the percentage of gap is, the more effectiveness of the algorithm posses. The percentage of gap of each instance is presented in Table 2.

It can be easily seen that our AIW-PSO algorithm for the OCST problem produces the most encouraging results for the tested benchmark instances. In Palmer series, the best known solutions are achieved from Hoang's and ours in Palmer6 and Palmer12, whereas, in Palmer's GA and Li&Bouchebaba's GA, the gaps of 2.3933% and 2.1510% are resulted for Palmer6, and the gaps of 13.0663% and 0.8588% are recorded for Palmer12. The best algorithm for the Palmer24 is the Li&Bouchebaba's GA. Although our algorithm is not the best for the Palmer24, our obtained result gets much closer to the best known solution with the gap of only 0.1379%. The comparison on the Palmer instances is graphically represented in Figure 4-1. In Raidl series, our algorithm produces the favorable results, especially in the Raidl50 and Raidl75, as can be seen in Figure 4-2. Our AIW-PSO algorithm produces the same best known solutions for Raidl10 and Raidl20, and the gaps of less than 1% for Raidl50 and Raidl75. In Berry series, the solutions obtained by our algorithm are the same as the best known solutions, while the Li&Bouchebaba's GA creates the gap of 0.9033% for Berry35u. According to the absence of the Palmer's GA and Hoang's PSO results on Berry35 and Berry35u, the comparison graph of the Berry series is drawn from only Li&Bouchebaba's GA and our AIW-PSO algorithm and is shown in Figure 4-3.

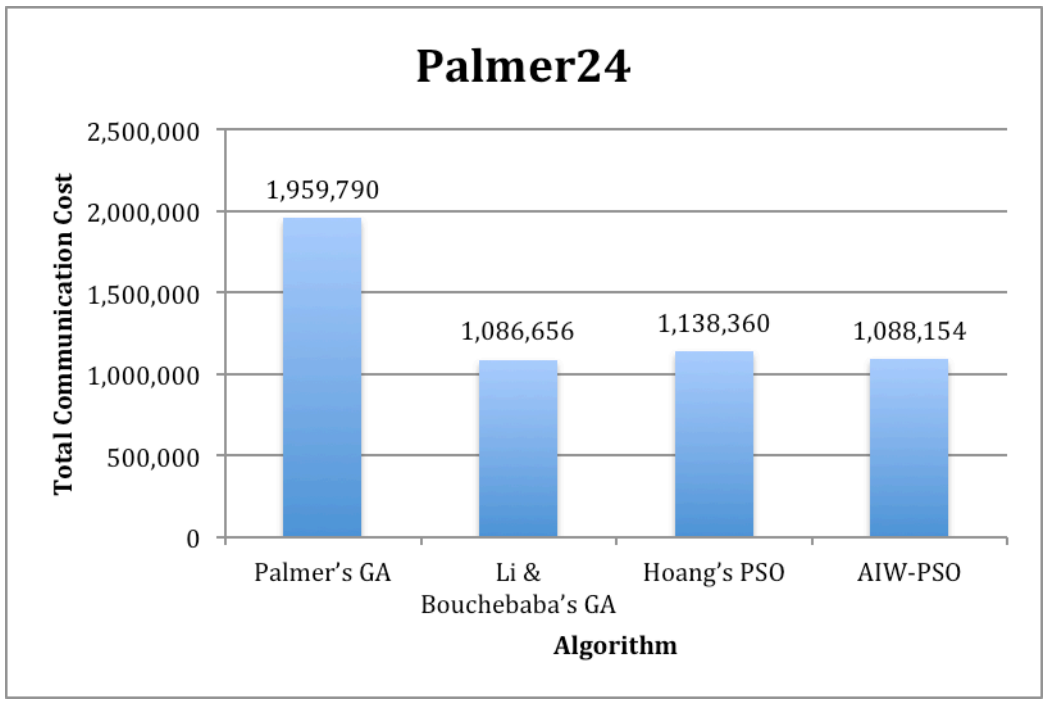

Figure 4-1: The best obtained solutions using four algorithms for Palmer24

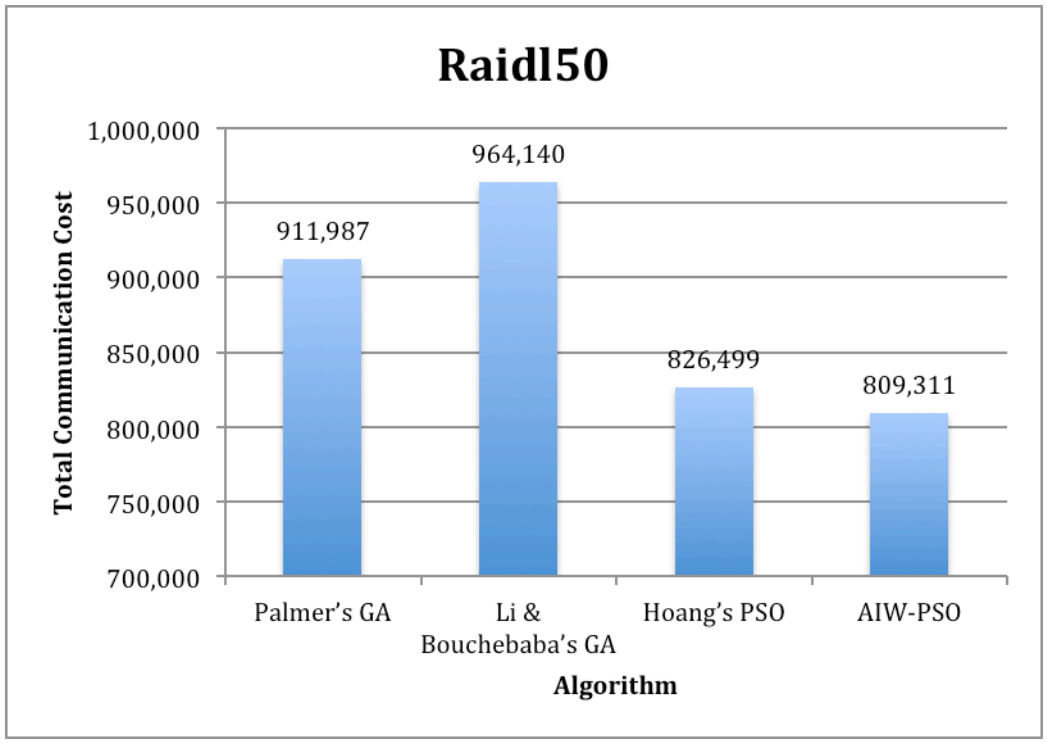

Figure 4-2: The best obtained solutions using four algorithms for Raidl50

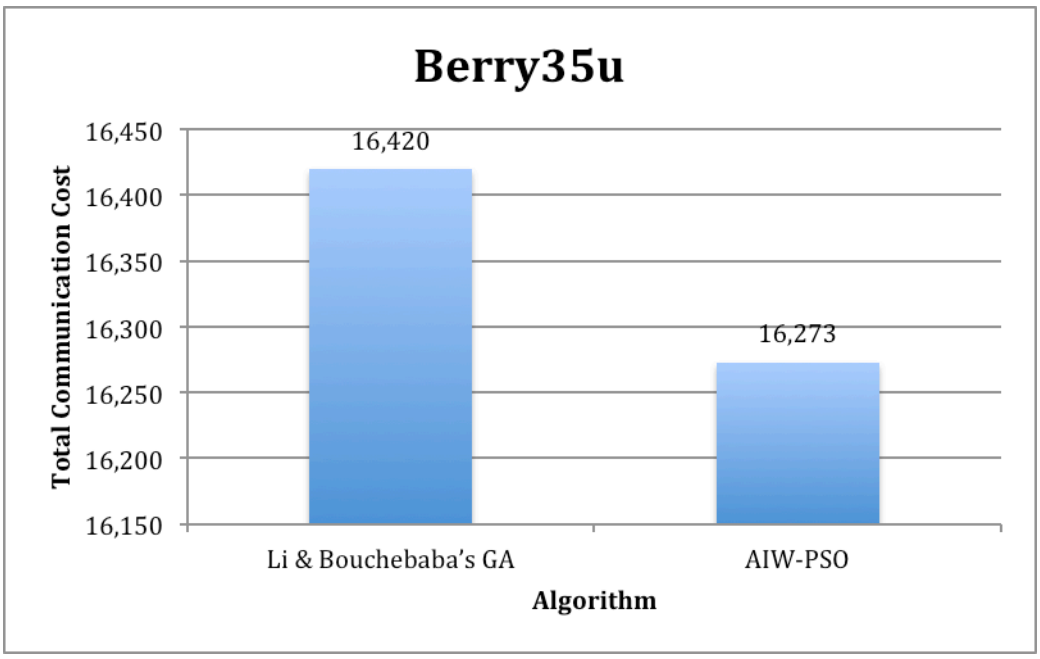

Figure 4-3: The best obtained solutions using Li&Bouchebaba's GA and our AIW-PSO for Berry35u

To see effect of the adaptive inertia weight on the PSO algorithm for the OCST problem, we show the comparison graph of the obtained solution over iterations from our standard PSO (with no adaptive inertia weight) and from the PSO with adaptive inertia weight (AIW-PSO). However, the standard PSO we implemented is not truly Hoang's PSO as some components of the algorithm are defined in our way because Hoang at al. had not given us such details. So, in this part of experiment, the comparison between our standard PSO with no adaptive inertia weight, which is called the O-PSO, and our proposed PSO with adaptive inertia weight (AIW-PSO) will be discussed.

Although we executed our algorithms on all ten OCST benchmark instances, some instances will be omitted when we do the comparison because its best known solution can be found in iteration 0 or iteration 1 as the convergence movement could not be represented from the run. Therefore, we show the comparison graph of Palmer24, Raidl50, and Raidl75 as in Figure 4-4, Figure 4-5, and Figure 4-6, respectively.

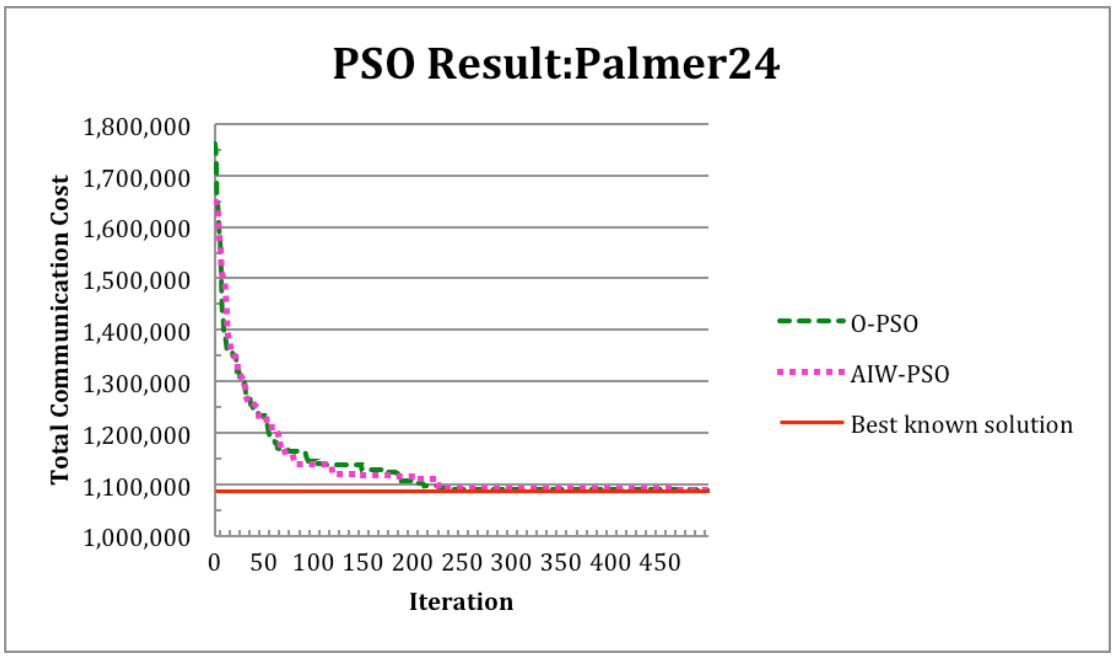

Figure 4-4: Comparison graph shows obtained solutions of Palmer24 over iterations between O-PSO and AIW-PSO

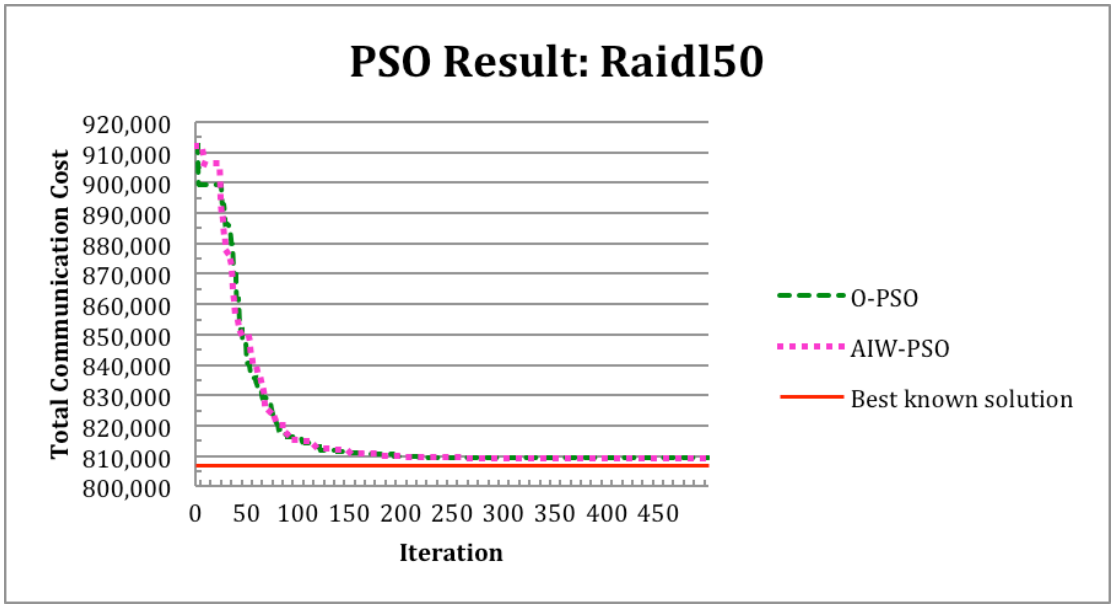

Figure 4-5: Comparison graph shows obtained solutions of Raidl50 over iterations between O-PSO and AIW-PSO

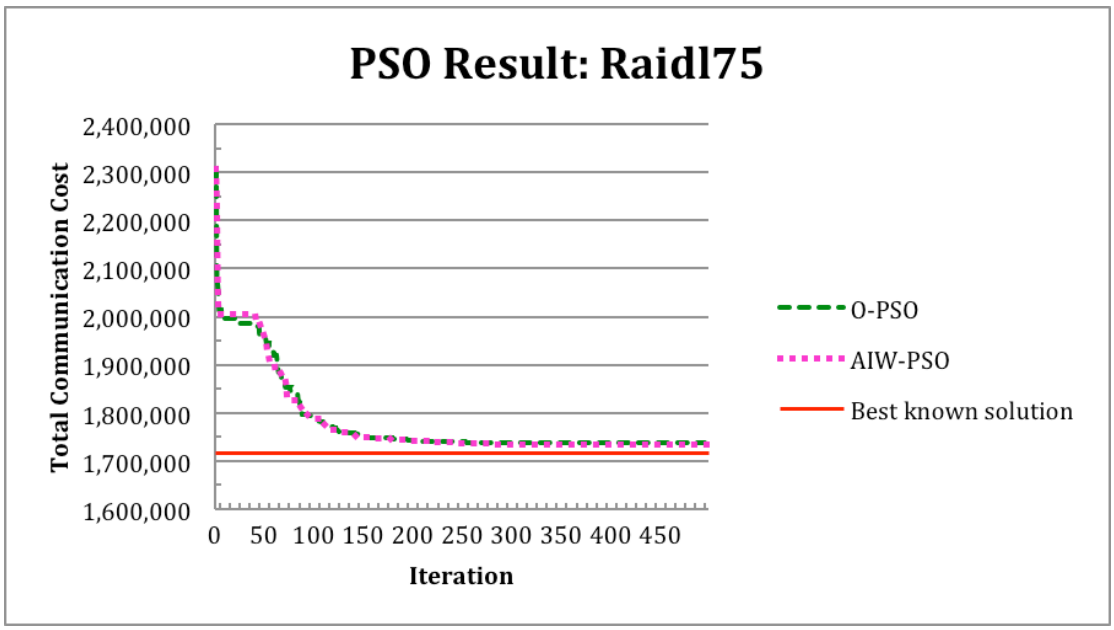

Figure 4-6: Comparison graph shows obtained solutions of Raidl75 over iterations between O-PSO and AIW-PSO

There is no major difference over iterations between O-PSO and AIW-PSO for the OCST problem. The graphs indicate that the adaptive inertia weights do not affect to the searching process with our designed parameter with population size of 1,000.

To avoid the population size effect, we conduct further experiments by reducing the population size to various values, i.e., 500, 100, 50 for the Palmer24, and 500, 100 for the Raidl75. We do not employ the population size of 50 to the Raidl75 as the population size must be greater than $(n+1)$. Results from this experiment are displayed in Figure 4-7 and Figure 4-8.

For the Palmer24 network, when population size is greater than 100, there is no significant difference in the obtained solutions among different population size setting. While the population size is reduced to only 50, the obtained solution is considerably far from the best know solution as can be seen in Figure 4-7. For the Raidl75 network, there is no critical variation when population size is greater than 500. When population size is reduced to 100, performance of the PSO algorithm is greatly dropped. Therefore, population size is extremely important to the final solution in PSO algorithm.

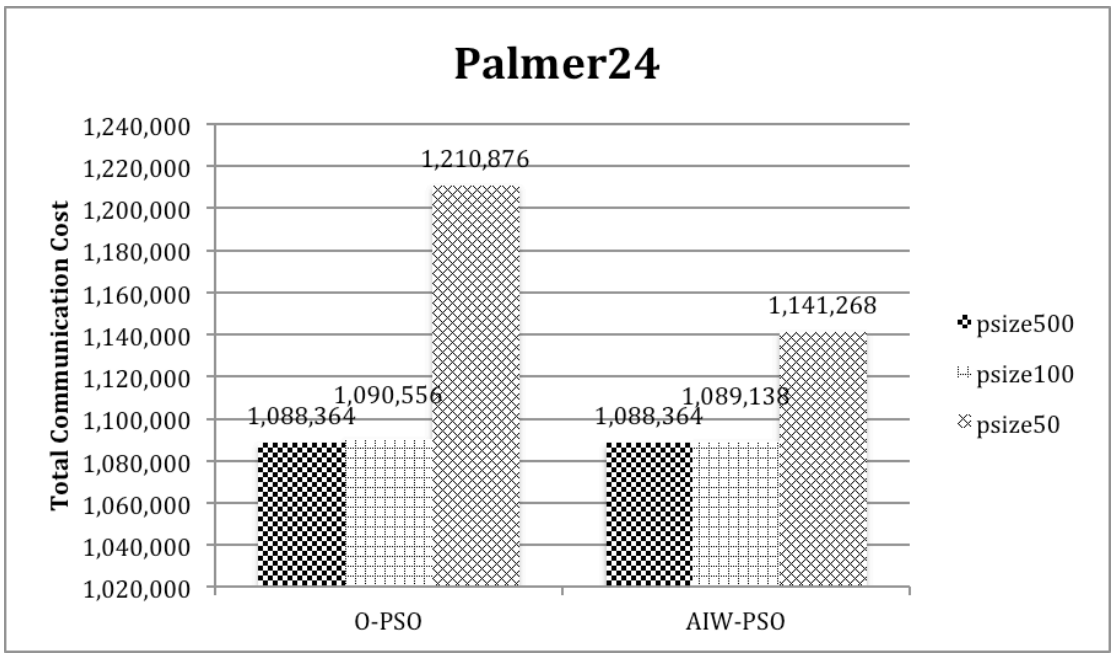

Figure 4-7: The obtained solutions from various population size settings of Palmer24

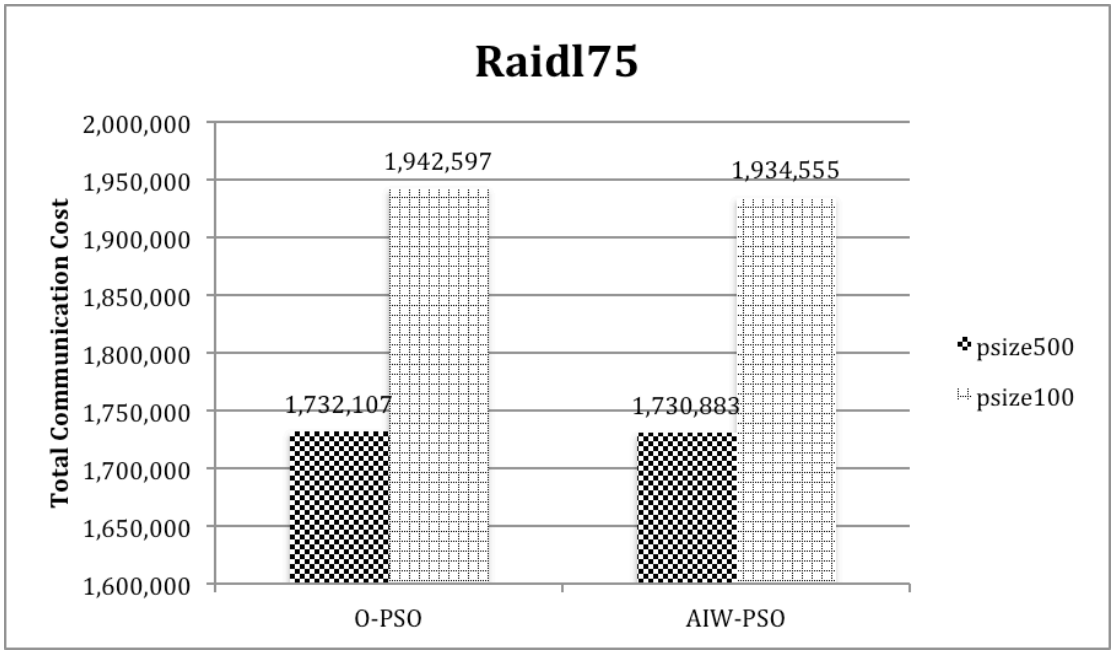

Figure 4-8: The obtained solutions from various population size settings of Raidl75

Accordingly, we reduce the population size of our AIW-PSO algorithm for the Palmer24 to only 50 and present the comparison graph in Figure 4-9.

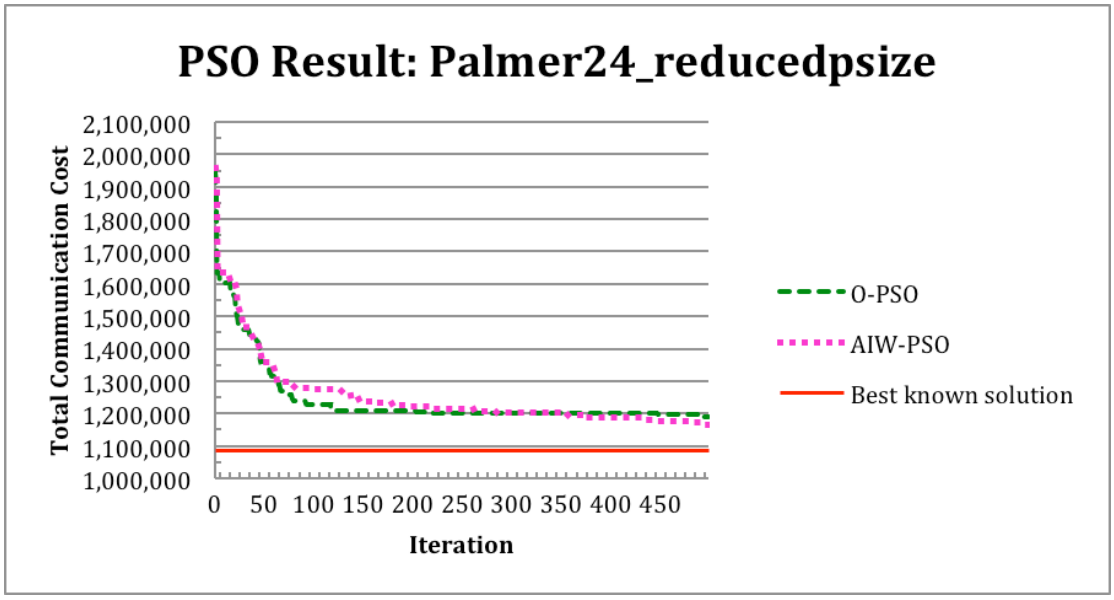

Figure 4-9: The obtained solution of Palmer24 over iterations with reduced population size

For Palmer24, O-PSO performs better than AIW-PSO until iteration 284 where AIW-PSO explores to the new global best region. Finally, the AIW-PSO produces better obtained solution.

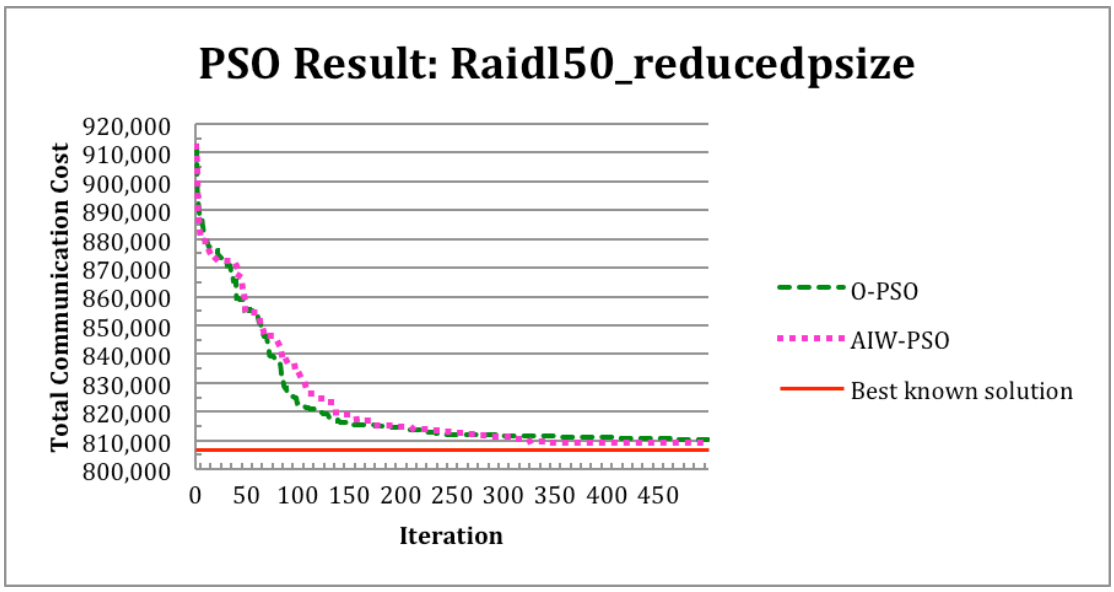

Figure 4-10: The obtained solution of Raidl50 over iterations with reduced population size

For Raidl50, the O-PSO obtained better solution after iteration 55 until iteration 304. Afterwards, the solution from the AIW-PSO is better. Accordingly, we can comprehend that when population size is large enough, adaptive inertia weights cannot show their effect in the optimal search. But when population size is reduced, the effect of the adaptive inertia weight is shown. Therefore, the beneficial of adaptive inertia weight depends on population size in the AIW-PSO algorithm.

In addition, we implement both O-PSO algorithm and AIW-PSO algorithm with Palmer24 network and compare the results with Hoang's initial pattern and with the new initial pattern with population size of 50. The average values of the obtained solutions are recorded and are used in comparison. Figure 4-11 shows that both PSO algorithms with new initial pattern produce better results than those with Hoang's initial pattern. Therefore, this shows advantage of the new initial pattern when population size is small.

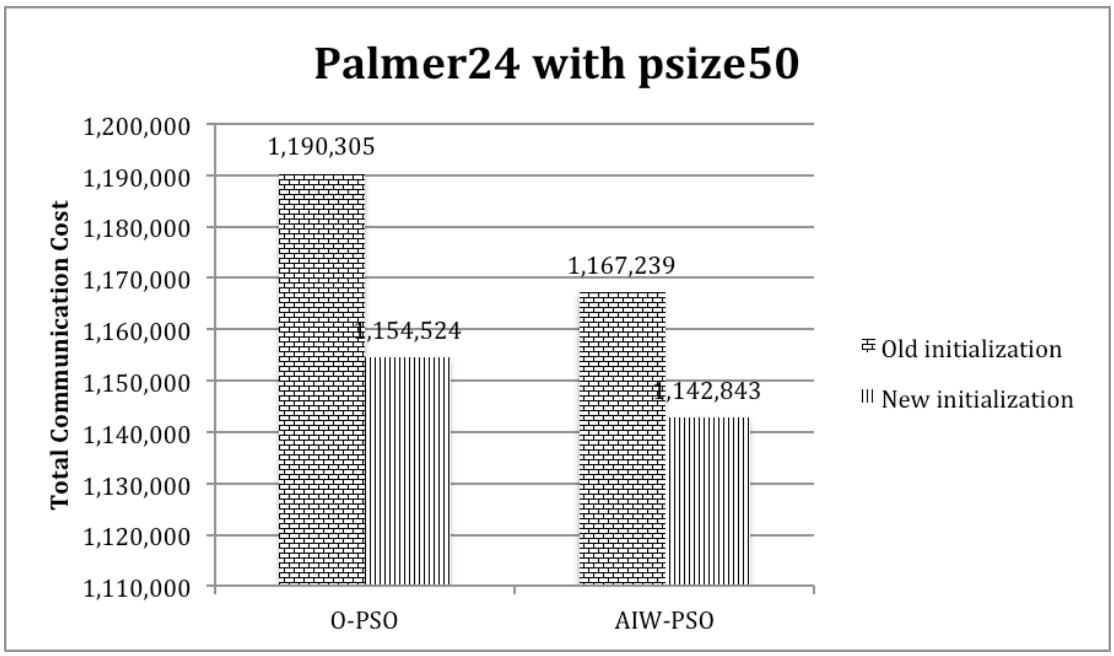

Figure 4-11: The average value of obtained solutions of Palmer24 from PSO with Hoang's initial pattern and with new initial pattern when population size is 50

In summary, both adaptive inertia weight and population initialization pattern for PSO algorithm have a significant impact to PSO algorithm only when number of population is small enough, otherwise, the effect of the population size dominates the search process.

Chapter V
Conclusion

We propose the improved version of the novel Particle Swarm Optimization-based algorithm for the OCST problem. The process of our main algorithm is based on Hoang et al.'s algorithm with the measure of dissimilarity between a pair of trees defined as in this paper. We include the adaptive inertia weight strategy and revise the particle initialization pattern, which affect the final solution of the particle search.

Our AIW-PSO algorithm for the OCST problem yields better results and our results assure our knowledge that initial population in heuristics-based algorithm have obvious impact to the obtained solution at final run. Furthermore, if we look further into the effect of our modifications, the adaptive inertia weight and the new initialization pattern affirm their benefit over the PSO algorithm when population size is small. Therefore, number of population in the PSO algorithm has a significant impact to the particle search.

The next step of our work is to test with more real world problems which fall into the OCST problem area.

References

[1] Ahuja, R. K., Murty, V. V. S. *Exact and Heuristic Algorithms for the Optimum Communication Spanning Tree problem.* Transportation Science, Vol.21, No.3, August 1987. (1987). pp. 163-170.

[2] Magnanti, T. L. and Wong, R. T. *Network Design and Transportation Planning: Models and Algorithms.* Transportation Science, Vol.18. (1984). pp. 1-55.

[3] Hu, T. C. *Optimum Communication Spanning Trees.* SIAM Journal on Computing 3(3). (1974). pp. 188-195.

[4] Johnson, D. S., Lenstra, J. K. and Kan, A. H. G. Rinnooy. *The Complexity of the Network Design Problem,* Networks, 8. (1978). pp. 279-285.

[5] Rothlauf, F. *Representations for genetic and evolutionary algorithms (2nd edition).* Springer. (2006).

[6] Contreras I, Fernández E, and Marín A. *The tree of hubs location problem.* European Journal of Operational Research, Volume 202, Issue 2, 16 April 2010. (2010). pp. 390–400.

[7] Contreras, I., Fernández, E., and Marín, A. *Lagrangean bounds for the Optimum Communication Spanning Tree Problem*. An Official Journal of the Spanish Society of Statistics and Operations Research (TOP), 18. (2010). pp. 140-157.

[8] Peleg, D. and Reshef, E. *Deterministic polylog approximation for minimum communication spanning trees.* Lecture notes in computer science, vol. 1443. Springer, Berlin. (1998). pp. 670–686.

[9] Wu, B. Y., Chao, K. M. and Tang, C. Y. *Approximation Algorithms for Some Optimum Communication Spanning Tree Problems.* Discrete Applied Mathematics, 102. (2000). pp. 245-266.

[10] Sharma, P. *Algorithms for the optimum communication spanning tree problem.* Annals of Operations Research, March 2006, Volume 143, Issue 1. (2006). pp. 203-209.

[11] Dionne, R. and Florian, M. *Exact and approximate algorithms for optimal network design*. Networks. Vol. 9. (1979). pp. 37-59.

[12] Palmer, C. C. *An approach to a problem in network design using genetic algorithms.* PhD Thesis. Polytechnic University, Computer Science Department, Brooklyn, New York. 1994.

[13] Soak, S. M., *A New Evolutionary Approach for the Optimal Communication Spanning Tree Problem*. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Volume E89-A Issue 10, October 2006, (2006). pp. 2882-2893.

[14] Fischer, T. and Merz, P. *A Memetic Algorithm for the Optimum Communication Spanning Tree Problem*. Lecture Notes in Computer Science, Volume 4771/2007. (2007). pp. 170-184.

[15] Holland, J. H., *Adaptation in Natural and Artificial Systems*, Cambridge, MA: MIT Press. Second edition (1992), (First edition, University of Michigan Press, 1975). 1975/ 1992.

[16] Hoang, A. T., Le, V. T., and Nguyen, N. G. *A Novel Particle Swarm Optimization – based Algorithm for the Optimal Communication Spanning Tree problem.* 2010 Second international Conference on Communication Software and Networks. (2010). pp. 232-236.

[17] Li, Y. and Bouchebaba, Y. *A new genetic algorithm for the optimal communication spanning tree problem.* Proceedings of Artificial Evolution: Fifth European Conference. Berlin, Springer. (1999). pp. 162–173.

[18] Rothlauf, F. *Design of Modern Heuristics.* Natural Computing Series. (2010). pp. 185-220.

[19] Nickabadi, A., Ebadzadeh, M.M., and Safabakhsh, R. *A novel particle swarm optimization algorithm with adaptive inertia weight*. Applied Soft Computing, Volume 11, Issue 4, June 2011. (2011). pp. 3658–3670.

[20] Kennedy, J. and Eberhart, R. *Particle swarm optimization*. Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, Vol. 4. (1995). pp. 1942-1948.

[21] Eberhart, R. and Kennedy, J. *A new optimizer using particle swarm theory*. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan. (1995). pp. 39-43.

[22] Palmer, C. C. and Kershenbaum, A. *Representing trees in genetic algorithms*. In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Date of Conference: 27-29 Jun 1994, vol.1. (1994). pp. 379-384.

[23] Raidl, G. R. and Julstrom, B. A. *Edge-Sets: An Effective Evolutionary Coding of Spanning Trees*. Evolutionary Computation, IEEE Transactions on, Date of Publication: June 2003, Volume 7, Issue 3. (2003). pp. 225- 239.

[24] Prim, R. C. *Shortest connection networks and some generalizations*. Bell System Technical Journal, 36. (1957). pp. 1389–1401.

[25] Shi, Y. and Eberhart, R., *A modified particle swarm optimizer*. Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, (1998). pp. 69-73.

[26] Engelbrecht, A. *Particle Swarm Optimization: Velocity Initialization*. 2012 IEEE Congress on Evolutionary Computation (CEC). Date of Conference: 10-15 June 2012. (2012). pp. 1- 8.

Appendix

Appendix

Pseudocode of AIW-PSO

```
/* Read input data */
Read n; // number of nodes in the original network
Read matrixD[i][j]; // distance matrix element
Read matrixR[i][j]; // requirement matrix element
/* Determine maximum distance in the distance matrix */
max_edge;
/* Determine degree of each node from minimum spanning tree (MST) from original distance */
PrimMST(matrixD) // find the MST using Prim's algorithm
/* Designed parameter */
population_size = 1,000;
knearest = 3; // for determining local best
maxIter = 500 // maximum number of iteration
/* Algorithm */
Initialize gbestValue = (max_edge)^5; // global best value
Initialize iter = 0;
Initialize w = 1; // inertia weight for the iter=0
For (k=1 to population_size) // Initialization step
        Initialize currentposition[k]; // vector size n
        Initialize currentvelocity[k] = 0; // vector size n
        Initialize particle[k] = currentposition[k];
        Initialize pbest[k] = particle[k];
EndFor
While (iter < maxIter)
        success_count = 0; // count of success(for computing inertia weight)
        para2 = 1; // the node-biased parameter for the modified matrix
        k = 0;
        While (k < population_size)
                /*Calculate fitness value of the particle 'k' */
                for i=1 to n
                        for j=1 to n
                        modifiedD[i][j] = matrixD[i][j] + (para2*(currentposition[k][i] +
currentposition[k][j])*max_edge);
                        EndFor
```

```
                EndFor
                PrimMST(k, modifiedD); // find the MST using Prim's algorithm
                Print TCC[k]; // Total communication cost of the MST
                fitness[k] = 1/TCC[k];
                /* If the fitness value is better than the best fitness value in history, then set the
current value as the new pBest */
                if (iter > 0)
                        if (fitness[k] > pbestValueCurrent[k])
                        success_count = success_count + 1;
                        pbestValueNew[k] = fitness[k];
                                For y=1 to n
                                        pbest[k][y] = currentPosition[k][y];
                                EndFor
                                If (TCC[k] < gbestValue)  // update global best value
                                        gbestValue = TCC[k];
                                        gbestIndex = k;
                                        gbestIter = iter;
                                        For y=1 to n // update global best position
                                                gbest[y] = currentPosition[gbestIndex][y];
                                        EndFor
                                EndIf
                        else pbestValueNew[k] = pbestValueCurrent[k];
                else    {
                        pbestValueNew[k] = fitness[k];
                        If (TCC[k] < gbestValue)  // update global best value
                                gbestValue = TCC[k];
                                gbestIndex = k;
                                gbestIter = iter;
                                For y=1 to n // update global best position
                                        gbest[y] = currentPosition[gbestIndex][y];
                                EndFor
                        EndIf
                        pbestValueCurrent[k] = pbestValueNew[k];
                        }
                EndIf
                k = k + 1;
        EndWhile
```

```
        /* Find percentage of success*/
        percentage_success = success_count/ population_size;
        w_min = 0;
        w_max = 1;
        w = ((w_max – w_min)*percentage_success) + w_min;
        /* Choose the particle with the best fitness value of all the neighbor particles as the lbest */
        For a=1 to population_size
                For b=1 to population_size
                        Compute farness[a][b]; // as described in subsection 3.3
                EndFor
        EndFor
        For a=1 to population_size
                findlocalbest(a, knearest, farness);
        EndFor
        /* Update particles */
        iter = iter + 1;
        For a=1 to population_size
                /* Calculate particle velocity */
                For r=1 to n
                        newvelocity[a][r] = w*currentvelocity[a][r] + (1*rand01*(pbest[a][r] -
currentposition[a][r])) + (1*rand02*(lbest[a][r] - currentposition[a][r]));
                EndFor
                /* Update particle position */
                For j=1 to n
                        newposition[a][j] = currentposition[a][j] + newvelocity[a][j];
                EndFor
        EndFor
EndWhile
```

Biography

Name: **Thakorn Chatchaisathaporn**

Date of birth: **26 September 1982**

Place of birth: **Bangkok, Thailand**

Education degree:
- Year 2005: **Bachelor of Science in Mathematics, Chulalongkorn University, Thailand**

Publication:
- 14-16 Mar 2013: 18th Annual Meeting in Mathematics, *IMPROVED PARTICLE SWARM ALGORITHM BY MINIMUM SPANNING TREE FOR THE OPTIMAL COMMUNICATION SPANNING TREE PROBLEM*