WATER DISTRIBUTION NETWORK OPTIMIZATION: A HYBRID APPROACH

By

XUEWEI QI

(Under the Direction of Ke Li)

ABSTRACT

An urban water distribution network (WDN) is a network of components (e.g. pipes, pumps, valves, tanks, etc.) that transport water from a source to the consumers. Due to the substantial cost associated with the material and installation of WDN, it is necessary to optimize its design by selecting the lowest cost combination of appropriate component configuration while the hydraulic and resilience constraints are satisfied. Thus far, a large variety of algorithms have been proposed for this optimization problem, among which swarm intelligence algorithms (SIA) attract the most recent attentions. In the project, several new SIAs are tested on this problem for the first time and different Machine Learning techniques are also used to further improve the performance of these swarm intelligence search algorithms. Ten different algorithms are proposed in this thesis project for WDN optimization problem. All of the proposed algorithms are tested on two famous benchmark networks and their performances are compared extensively, the results show that some of the proposed algorithms are very promising in the real application, especially for large size water distribution networks. What is more, one of the proposed

algorithms successfully achieves a new record of the best solution cost on the larger size network.

INDEX WORDS: Swarm Intelligence Algorithms, Machine Learning, PSO, WDN, Engineering Optimization, FSS, EDA, EM

WATER DISTRIBUTION NETWORK OPTIMIZATION: A HYBRID APPROACH

By

XUEWEI QI

B.E., China Agricultural University, Beijing, China, 2007

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2013

WATER DISTRIBUTION NETWORK OPTIMIZATION: A HYBRID APPROACH

By

XUEWEI QI

Major Professor: Ke Li

Committee: Walter D. Potter

Khaled M. Rasheed

Electronic Version Approved:

Maureen Grasso

Dean of the Graduate School

The University of Georgia

August 2013

ACKNOWLEDGEMENTS

At the very beginning, all my gratitude would absolutely be first given to my parents: Qiyang Qi and Bifen Xiong. Without their support, I could not make any achievement so far.

Of course, this thesis would not have been completed without the efforts of following individuals who have worked with me in UGA:

First and foremost, my sincere thanks go to my major advisor Dr. Ke Li for his mentoring and providing me the funding and equipment for my research projects. Dr. Li also has been very generous with his time and wisdom for providing me advice on how to do research and how to write papers. His attitude towards life as well as scientific research always inspires me a lot. The research work under his guidance in The University of Georgia will be a valuable and beneficial experience in my life and also a good foundation for my future academic career.

Besides, I also would love to show my sincere thanks to Dr. Walter D. Potter and Dr. Khaled M. Rasheed. As my committee members, they helped me a lot with great kind, patience and also their outstanding expertise since the very beginning of my study in UGA. I really appreciate their support and guidance in my graduate study and research. Most of the basic ideas of my thesis project derive from the experience obtained in their courses. It is them who led me into the realm

of Computational Intelligence, where I found great research interests and formed my research direction. The research method and academic attitude I learned from them will definitely benefit me all of my future life.

In addition, I could not go without acknowledging my lab-mates, specifically including Fang Zeng, Pretthi Rao , Junjie Hou, , Xiang Li, Nathan Hester, and so on. They all made contributions to this thesis research, tremendously helpful and supportive.

Last but not least, sincere thanks to all of my friends, especially including Jun Chen, Lvjun Zhou, Timothy Gary, Shu Zhang, Xi Jiang and so on. Great deals of credits are owed to their support and encouragement in developing this technical report.

At last, I should also appreciate US. National Science Foundation and China Scholarship Council of Ministry of Education of China, which both have funded me for the research in UGA.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1 INTRODUCTION

1.1 Swarm Intelligence Algorithms

Evolutionary computation (EC) [1] is a kind of optimization methodology inspired by the mechanisms of natural evolution and behaviors of living organisms. Generally speaking, EC algorithms include genetic algorithm (GA), evolutionary programming (EP), evolutionary strategies (ES), genetic programming (GP), learning classifier systems (LCS), differential evolution (DE), and estimation of distribution algorithm (EDA). Recently, by interpreting and modeling swarm intelligence, a new category of evolutionary algorithm: "Swarm Intelligence Algorithm (SIA)" is identified and has gained increasing popularity in the EC research community [2].

Swarm intelligence (SI) is a type of intelligence, which is observed in the collective behavior of decentralized, self-organized swarm systems, especially biological swarm systems in the nature.

As a sub area of evolutionary algorithms (EA), SIA focuses on the study of computational systems inspired by the "collective intelligence", which emerges through the cooperation of large numbers of homogeneous agents in certain environment. In a swarm, agents often follow similar

simple rules, and interact locally with each other in the local environment. Most of the original inspirations come from natural biological systems, such as insects and animal swarms. Typical examples are bird flock, fish school, and ant colony. Like other EAs, such swarm intelligence algorithms or strategies are typically applied to search and optimization domains. The most popular examples are particle swarm optimization (PSO), ant colony optimization, fish school search and etc.

1.2 Use Machine Learning to Improve Evolutionary Algorithms

Machine Learning, a branch of artificial intelligence, is about the construction and study of systems that can learn from data. Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm or the type of input available during training the machine: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning.

As the development of both Machine Learning (ML) and evolutionary algorithms, many researchers have turned their attention to combining these two different type of methods so that they can complement with each other with their own advantages. There is a large variety of ways on how to hybridize them together, among which using ML to enhance EAs is a very active research spot recently [3].

A large variety of Machine Learning (ML) techniques have been used to enhance the Evolutionary algorithms [1]. These ML techniques include: statistical methods, interpolation and regression, clustering analysis, orthogonal experimental design, opposition-based learning, artificial neural networks, support vector machines, case-based reasoning, reinforcement

learning, and competitive learning and Bayesian network. In terms of the way ML used to improve EAs, there are 5 major different directions:

1. Population Initialization: ML techniques could be used to create or improve the initial population for EAs.
2. Fitness Evaluation and Selection: ML techniques could be used to model and approximate the fitness evaluation functions which are usually the most computational expensive part of the search algorithms.
3. Population Reproduction: ML could be used to predict promising region or reduce dimensionality of the problem.
4. Algorithm Adaptation: Use ML to adapt both Parameters and Operators.
5. Local Search: Use ML to control or perform local search operators.

1.3 Optimization of Urban Water Distribution Network

An urban water distribution network (WDN) is a network of components (e.g. pipes, pumps, valves, tanks, etc.) that transport water from a source (e.g. reservoir, treatment plant, tank, etc.) to the consumers (e.g. domestic, commercial, and industrial users). Due to the substantial cost associated with the installation and material of WDN, it is necessary to optimize its design by selecting the lowest cost combination of appropriate component sizes and component settings while the hydraulic and resilience constraints are satisfied. In engineering practices, the diameter setting of pipelines is the major factor which determines the size of components and the installation cost. Therefore, the selection of pipe diameters, namely "pipe sizing", becomes the classical WDN optimization challenge. The objective of pipe sizing is to minimize the total cost of the network design. Mathematically, this optimization problem can be defined as a non-linear,

non-convex and multi-modal problem or an NP-hard combinatorial problem, involving a complex set of implicit constraints, such as conservation of mass and energy equations, which are commonly satisfied through the use of hydraulic simulation solvers. Thus far, a variety of optimization methods have been proposed for WDN optimal design, including classical operational research techniques (e.g. linear programming); typical evolutionary algorithms (e.g. genetic algorithm) and relatively new swarm intelligence algorithms (e.g. ant colony optimization). Through the literature review, there is an obvious trend can be observed, which that is more and more researchers are shifting their focus onto the new swarm intelligence algorithms for this hard optimization problem.

1.4 Organization of This Thesis

In this thesis research, two famous swarm intelligence algorithms and their variants are introduced to solve the WDN optimal design problem, they are all tested on the benchmark networks and their performances are compared with each other experimentally.

The remainder of this thesis is organized as follows: Chapter 2 reviews the related work and provides an elaborate analysis of the previous work. Chapter 3 provides the motivation and the objectives of this thesis research. Chapter 4 describes the problem representation and benchmark examples. Details of conventional version of the selected swarm intelligence algorithms are given in Chapter 5. Chapter 6 explains what kind of Machine Learning techniques and how they are used to further improve the performance of the selected search algorithms. The last two sections compare the performances of different algorithms on the specific WDN optimization problem, make a conclusion and point out the potential future work. For the sake of convenience to the readers, following table 1.1 provides an overall sense of the structure of this thesis:

Table 1.1.Overview of the Structure of This Thesis

Chapter	Total Number of Pages
1. Introduction	6
2. Literature Review & Analysis	5
3. Motivation & Objectives	1
4. Problem Representation	6
5. Swarm Intelligence Algorithms	9
6.Machine Learning enhanced Search Algorithms	68
7. Performance Comparison	5
8. Conclusions & Future Work	3

CHAPTER 2

2 LITERATURE REVIEW AND ANALYSIS

Pipe sizing is one essential step in the optimal design of WDN, where the decision variables have primarily been associated with the pipes within the system. More specifically, the decision variables in pipe sizing are limited to the selections of diameters of the pipes in a WDS and the only hydraulic constraint in these optimization process are the minimum allowable pressures at each of the nodes that have to be satisfied.

Mathematically, this optimization problem can be defined as a non-linear, non-convex and multi-modal problem or an NP-hard combinatorial optimization problem, involving a complex set of implicit constraints, such as conservation of mass and energy equations, which are commonly satisfied through the use of hydraulic simulation solvers.

In the past three decades, a large variety of optimization methods have been proposed for WDN pipe sizing by many researchers. These methods can be generally classified into three distinct major categories:

2.1 Deterministic Methods

At the very early stage of this research, many earlier researchers have attempted to solve this optimization problem using some traditional operational research techniques (including linear programming, dynamic programming, and nonlinear programming). But no efficient method is

found due to the limitations they intuitively have. These models usually result in a local optimum which is dependent on the starting point in the search process. For more information, the readers are referred to [4]~[11].

2.2 Evolutionary Algorithms

In the optimization domain, evolutionary algorithms are one of the most popular algorithms recently. Many researchers shifted the focus from traditional operational methods to the evolutionary algorithms and a number of EAs have been introduced to solve the WDS optimization problem, such as Genetic Algorithms ([12]~[21]), Particle Swarm Optimization([22]~[23]), Simulated Annealing[24], Tabu Search[25], Cellular Automata[17], Harmony search[26], Frog Leaping Algorithm[27], Honey-Bee Mating Optimization[28], Immune Algorithm[29]. Shuffled Complex Evolution[30], Ant Colony Optimization[31], Genetic Heritage Evolution[32]. In table 2.1, a full list of evolutionary algorithms, which have been used, is given.

Table 2.1.Evolutionary Algorithms Applied in WDN Optimization.

Algorithms	**Year**
Genetic Algorithm	1987, 1996, 1999, 2001, 2005, 2006, 2008, 2010(3), 2012
Simulated Annealing	1999
Tabu search	2004
Shuffled Complex Evolution	2004
Cellular Automata	2006
Ant Colony Optimization	2006 *
Particle Swarm Optimization	2008, 2010,2012 *

Immune Algorithm	2008
Harmony Search	2009
Genetic Heritage Evolution	2010
Frog Leaping Algorithm	2010 *
Honey-Bee Mating Optimization	2010 *
Memetic Algorithm	2010

Among these evolutionary algorithms, swarm intelligence algorithms attract the attentions of many researchers, including Particle Swarm Optimization, Ant Colony Optimization and Honey-Bee Mating Optimization (marked with * in table 1). This category of evolution algorithms share some similar characteristics and rationales. They are more appropriate for problems with high dimensionality. In addition, new swarm intelligence algorithms are still being created by observing and molding the collective behavior of different biological swarms in the nature. Therefore, applying new swarm intelligence algorithms for the optimal design of WDN is still a hot and promising research topic.

2.3 Hybrid Strategies

More recently, a new trend to improve or create novel efficient search algorithms is to hybridize different algorithms with different search abilities. As we know, some evolutionary algorithms (e.g. Genetic Algorithm) are good at exploring the search space while others (e.g. Iterated Local Search) performs better in exploiting the local search space. The hybrid algorithms attempt to obtain the best from the hybridization of classical evolutionary search algorithms that perform

together and complement each other to produce a new efficient algorithm modal. R. Banos [33] proposes a Memetic Algorithm which applies a local search process to each of the agents in the iteration process of an outer population based meta-heuristic algorithm to optimize the pipe sizing problem. The experimental results show that the proposed hybrid algorithm outperforms other methods in comparison, especially when the network size increases. As this is still an active research area, there is still large space for us to improve the efficiency of the optimization algorithms and design promising hybrid strategy for real-world engineering applications.

2.4 Swarm Intelligence Algorithms

As we can see in table 2.1, the methods marked with "*" are from the same category of evolutionary algorithms called "swarm intelligence algorithms". This is a new trend in the development of new optimization tool for water distribution network. The previous research already shows some of their advantages over other categories of evolutionary algorithms, such as fast convergence rate and high efficiency. Since there are many other swarm intelligence algorithms have not been tested in this specific engineering optimization problem, it is still a promising research area.

2.5 Limitations of Previous Work

Although all the above reviewed methods have shown their successful application on the optimization of water distribution network, there are still some limitations for these methods:

1. No algorithm could be regarded as perfect tool for optimization of water distribution. This is due to the fact that there is no big difference between them in terms of the overall performance and also the fact that some algorithms are better in some aspect of the

performance (e.g. efficiency), while they are worse than other in other aspect of the performance (e.g. reliability).

2. Lack of generality. Most of the algorithms are only tested on the small size benchmark networks, not tested on the large size network. Since a good performance on a small size network cannot guarantee good performance on larger size network.

3. Specifically, for the research on swarm intelligence algorithms, only a limited number of swarm intelligence algorithms have been tested on this engineering optimization problem. What is more, few variants of these algorithms have been proposed for the optimization problem.

CHAPTER 3

3 MOTIVATIONS & OBJECTIVES

3.1 Motivation

Inspired by above analysis of the literature and discussion of limitations of previous work, there are three major directions of further research on WDN optimization problem:

1) Continue to introduce new algorithms (e.g. swarm intelligence algorithms) that have never been used for this problem and test them on the benchmark networks;
2) Further improve the performance of existing methods and propose more effective variants of these algorithms.

3.2 Objectives

The overall goal of this proposed project is to propose new efficient search algorithms for optimization of WDN and try to achieve better optimal solution than the literature on the benchmark example networks. The specific objectives are as follows:

1. Apply new Swarm Intelligence optimization algorithms (e.g. Fish School Search) and their new variants to the optimal design of WDS and try to achieve better optimal solutions on the benchmark networks.
2. Use different Machine Learning methods to further improve the performance of the proposed optimization algorithms and test them on the benchmark networks.
3. Compare different swarm algorithms experimentally.

CHAPTER 4

4 PROBLEM REPRESENTATION

4.1 Problem Formulation

The optimal pipe sizing for a water distribution network with a pre-specified layout can be described as:

$$\text{Minimize } C_0=\sum_1^N C_i L_i \quad (4.1)$$

Where L_i is the length of the link i; C_i is the cost per unit length of the pipe used in link i; N is the number of the links (pipes) used in the network; The above minimization is subject to the following constraints:

Hydraulic constraints:

$$\sum_{in(k)} q_i - \sum_{out(k)} q_i = Q_k, \quad k=1\ldots..J, \quad (4.2)$$

$$\sum_{i\in l} J_i = 0, \quad l=1,\ldots.L, \quad (4.3)$$

$$q_i = K ch_i d_i^{\partial} (J_i/J_i)^{\beta} \quad (4.4)$$

where J and L are the number of existing nodes and loops in the network respectively; q_i is the flow rate in pipe i; Q_k is the required demand at consumption node k; J_i is the head loss in the ith pipe; ch_i is the Hazen-Williams coefficient for the ith pipe and α= 2.63, β= 0.54, and K = 0.281 for q in cubic meters and d in meters. These constraints, therefore, describe the flow continuity at nodes, head loss balance in loops and the Hazen-Williams equation.

Head constraints:

$$H_{min} \leq H_k \leq H_{max}, \qquad k=1\ldots.J \qquad (4.5)$$

Pipe size availability constraints:

$$d_{min} \leq d_i \leq d_{max}, \qquad i=1\ldots..N \qquad (4.6)$$

where N is the number of existing pipes; H_{min} is the nodal head; H_{min} and H_{max} are minimum and maximum allowable nodal head; and d_{min} and d_{max} are minimum and maximum commercially available pipe diameters.

4.2 Representation & Fitness Evaluation

In the practical implementation, for the population-based algorithms, each individual is encoded as a string of integers, these integers represent the index of available commercial diameters, and table 4.1 gives an example.

Table 4.1.Individual Representation Example (n is the number of pipes)

Pipe1	Pipe 2	Pipe 3	……..	Pipe n-1	Pipe n
7	1	6	…….	4	5

Normally, pipes with larger diameters are more expensive. Larger diameters are given larger index in the algorithm.

As we are trying to minimize the total cost of the pipes in the network, it is easy to obtain following basic fitness function:

$$C_0 = \sum_1^N C_i L_i \qquad (4.7)$$

where C_i is the unit price of ith pipe and L_i is the length of ith pipe. However, this simple fitness function fails to include the hydraulic consideration into the fitness evaluation. Therefore,

a penalty method is used to formulate the optimization of a pipe network as an unconstrained optimization problem in which head constraints are included in the objective function leading to a new problem defined by minimization of the following penalized objective function:

$$\text{Minimize } C_0 = \sum_1^N C_i L_i(1+HD^2) \quad (4.8)$$

$$\text{HD}=\sum_{i\in N} \max(H_{i,min} - H_{i,actual}) \quad (4.9)$$

Where the $H_{i,min}$ and $H_{i,actual}$ denote the required minimal head pressure and actual head pressure in node *i* respectively. Total cost will be increased by means of the hydraulic head deficit HD. Although the mathematical calculations of the hydraulic head deficit of a pipe network are very complicated and time consuming, an existing hydraulic solver could be adopted to implement the calculation for the fitness function: EPANET2.0. This hydraulic solver could be downloaded freely from the following website: http://www.epa.gov/nrmrl/wswrd/dw/epanet.html

Figure 4.1 shows the general flow chart for pipe size optimization. One candidate solution is a set of diameters for all the pipes in the water distribution network. A famous hydraulic simulator (EPANET2.0) is used to evaluate each candidate solution. The input of the simulator is one candidate solution (diameter set) and output is the actual head pressure of each node in the network.

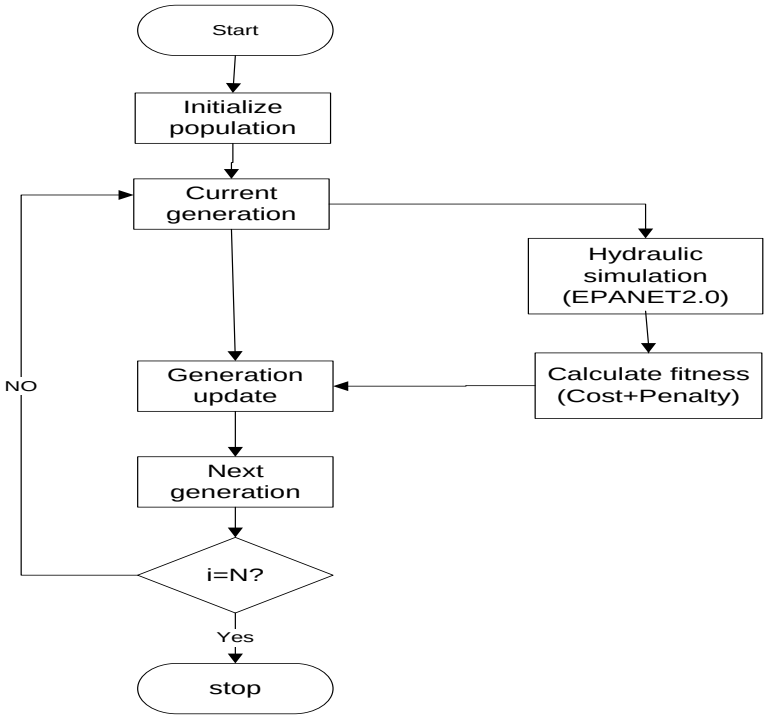


Figure 4.1.Schematic flowchart of the optimization algorithm for WDN

4.3 Benchmark Examples

In most of the literature, the algorithms are evaluated on several benchmark examples. In this project, all the proposed algorithms are tested on the following two famous benchmark examples:

4.3.1 Hanoi Network

The Hanoi network (Figure 4) presented by Fujiwara and Khang [34], requires the optimal design of 34 pipes, allowing a minimum hydraulic head of 30 meters for all its 32 nodes, by

means of 6 available diameters. The total solution space is then equal to 6^{34}. It serves as a prototype of medium sized network for the evaluation of optimization algorithm.

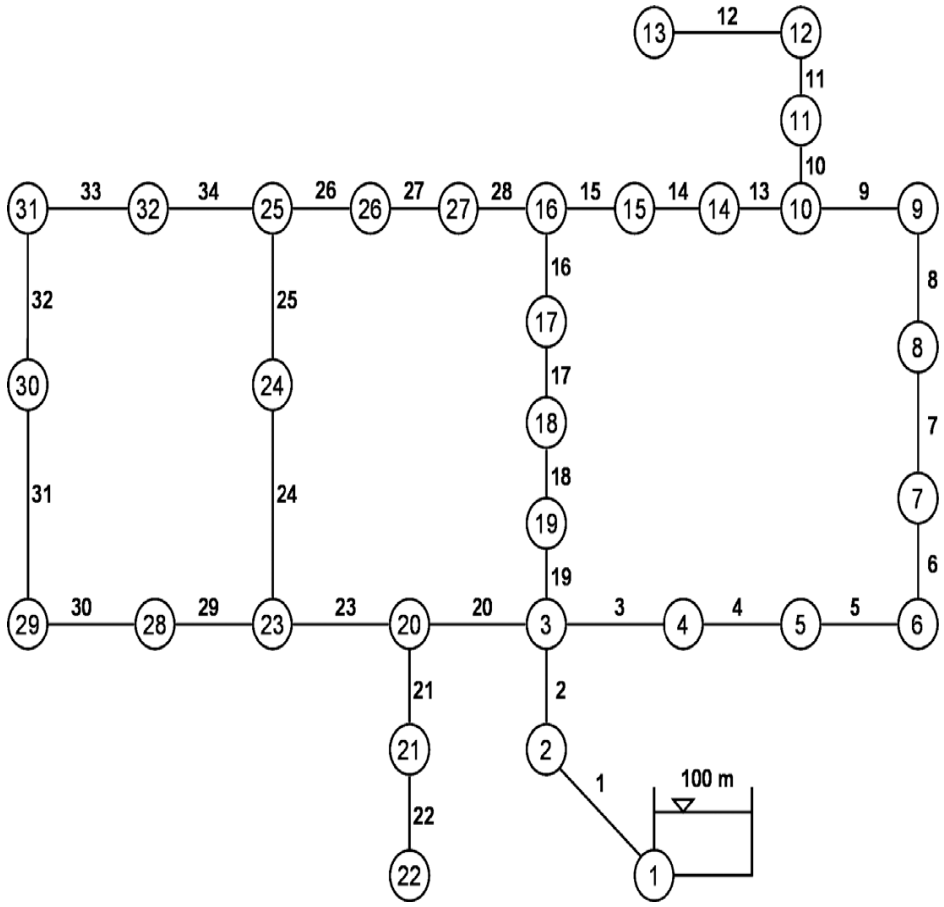


Figure 4.2Hanoi Network([34])

4.3.2 Balerma Network

Balerma network was originally proposed by Reca and Martinez [35]. It has a total of 443 demand nodes supplied by 4 source nodes (Figure 5). There are 454 pipes, arranged in 8 loops. The diameter of available pipes ranges from 125 and 600 mm with an absolute roughness coefficient k = 0.0025 mm. Total enumeration of the search space is 10^{10454}. The Darcy–Weisbach equation has been adopted to calculate head losses, using EPANET2. The minimum

required pressure head is 20 meters for each node. The Balerma network serves as a large sized WDN prototype for optimization algorithm evaluation.

Figure 4.3.Balerma Network ([35])

CHAPTER 5

5 SWARM INTELLIGENCE ALGORITHMS

In previous sections, we know that one major task for this thesis research is to introduce new swarm intelligence algorithms and test their performance on the benchmark examples. Two typical algorithms: Particle Swarm Optimization and Fish School Search are selected here. The reason why PSO is selected is that it is very popular in many optimization domains and it has already been successfully used for the WDN optimization problem. More importantly, there is still space for improvement of PSO on the problem because there are various ways to improve PSO. The reason why I select FSS is that it is a relatively new swarm intelligence algorithm and it has been proved successful in many difficult optimization problems. The details of the implementation of these two famous swarm intelligence algorithms are given in following sections.

5.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based optimization method inspired by the collective behaviour of a bird flock. It is an intelligent computational technique proposed by Kennedy and Eberhart in 1995[36]. This technique is commonly used to solve optimization problems of nonlinear functions. The idea behind PSO is to create particles that simulate the movements of birds to achieve a specific goal within the search space. It explores the social

behaviour of an organized group of individuals and the group's communication capacity. Each particle represents a solution in a multi-dimensional space. All the particles in the swarm use the same communication mechanism. In the most common PSO implementations, particles move through the search space using a combination of the attraction to the best solution of the entire swarm (gbest), and the attraction to the best solution that any particle in the neighbourhood has ever found (pbest). More specifically, each particle in the flock holds the following information:

(i) The current position x_i

(ii) the current velocity v_i

(iii) The best position that the particle has achieved so far Pbest_i

(iv) The best position that all the particles in the swarm has ever achieved gbest_i

During each iteration, each particle updates its position toward the Pbest_i and gbest_i according to the following equations:

$$v_{ij}^{t+1}=wv_{ij}^{t}+c_1r_{1j}(pbest_{ij}^{t}-x_{ij}^{t}) + c_2r_{2j}(gbest_{ij}^{t}-x_{ij}^{t}) \quad (1)$$

$$x_{ij}^{t+1}=x_{ij}^{t} + v_{ij}^{t+1} \quad (2)$$

Where j denotes the index of dimension and i is the number of particles; w is the inertia weight and t is the iteration number; r_1 and r_2 are two random numbers uniformly distributed in the range [0,1]; c_1 and c_2 are the acceleration factors. After all the particles are updated, the Pbest_i and gbest_i are also updated if better results are found. Both c_1 and c_2 are set to be 2 and w is set to be 0.8 in all the following experiments.

In order to obtain a performance benchmark for the following experiments on the improvement of PSO, we tested the original integer PSO (IPSO) on the benchmark examples first.

5.1.1 Experimental results of standard integer PSO

According to the inherent characteristic of WDN optimization problem, we choose to use integer PSO because there are limited number of available diameters for each pipe and these options could be encoded into integers. In order to obtain a benchmark for the following new variants of PSO, we tested this standard integer PSO on the network examples. The experimental setting and results are listed in table 5.1.

Table 5.1.Experimental Setting and Results of IPSO

Network	Population size	#of iteration	c1 c2 w	# of runs	Best cost	Average cost
Hanoi	500	500	2,2,0.8	30	$6.369x10^6$	$7.110x10^6$
Balerma	1000	1000	2,2,0.8	30	$6.299x10^6$	$6.729x10^6$

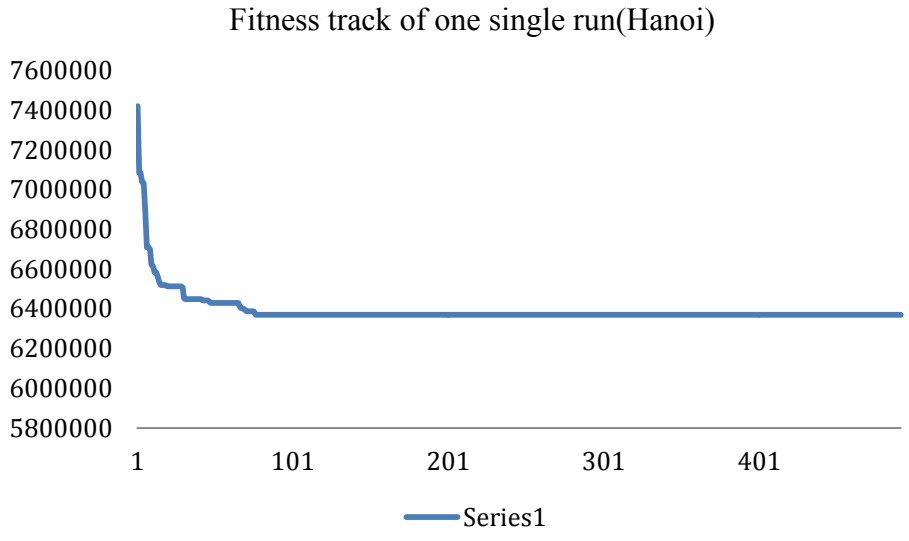


Figure 5.1. Fitness track on Hanoi network(IPSO)

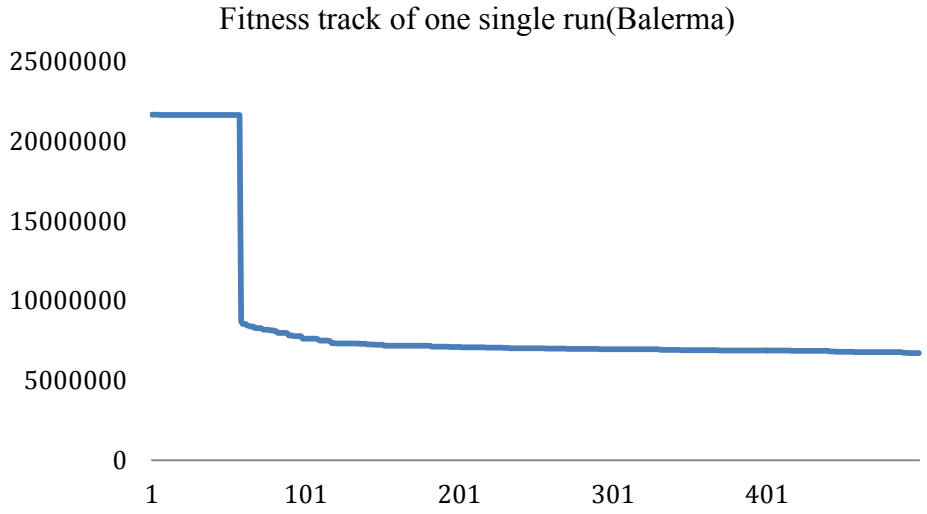


Figure 5.1.Fitness track on Balerma network(IPSO)

5.2 Fish School Search Algorithm

Many oceanic fish species, as with other animals, present social behaviour. There are two major categories of behaviours are observed in fish schools and they are also the main inspirations for modelling [37]:

- Feeding: inspired by the natural instinct of individuals (fish) to find food in order to grow strong and to be able to breed. Food here is used as a metaphor for the evaluation of candidate solutions in the search process. An individual fish can lose as well as obtain weight, depending on the regions it swims in;

- Swimming: it is the most obvious behaviour of a fish school and also the most sophisticated movement for modelling. There are different types of swimming movement and various motivations for these moves. It could be a self-motivated individual

movement or swarm-motivated collective movement. Swimming movement is primarily driven by feeding needs.

Inspired by aforementioned observations, a novel search algorithm Fish School Search (FSS) is proposed by Bastos Filho et al.[38]. And a set of operators are proposed to form the main routines of the Fish school search algorithm (FSS). To understand the operators, a number of concepts need to be defined. The concept of food is related to the function to be optimized in the process. For example, in a minimization problem the amount of food in a region is inversely proportional to the function evaluation in this region. The "aquarium" is defined by the delimited region in the search space where the fish can be positioned.

The operators are grouped in the same manner in which they were observed when drawn from the fish school. They are as follows:

- Feeding: food is a metaphor for indicating to the fish the regions of the aquarium that are likely to be good spots for the search process;
- Swimming: a collection of operators that are responsible for guiding the search effort globally towards subspaces of the aquarium that are collectively sensed byall individual fish as more promising with regard to the search process.

5.1.1 Feeding Operator

As in real situations, the fish of FSS are attracted to food scattered in the aquarium in various concentrations. In order to find greater amounts of food, the fish in the school can move independently (see individual movements in the next section). As a result, each fish can grow or diminish in weight, depending on its success or failure in obtaining food. It is believed that fish's weight variation is proportional to the normalized difference between the evaluation of fitness

function of previous and current fish position with regard to food concentration of these spots. The assessment of 'food' concentration considers all problem dimensions, as shown in the following equation (5.1):

$$W_{t+1}^{i}=W_{t}^{i}+\frac{f(X_{t+1}^{i})-f(X_{t}^{i})}{\max\{|f(X_{t+1}^{i})-f(X_{t}^{i})|\}} \quad (5.1)$$

where W_t^i is the weight of the fish i; X_t^i is the position of the fish i and f (X_t^i) evaluates the fitness function (i.e. amount of food) in X_t^i. Fish weight variation is evaluated once at every FSS cycle. An additional parameter, named weight scale (*Wscale*) was created to limit the weight of a fish. The fish weight can vary between"1" and *Wscale*. All the fish are born with weight equal to *Wscale*.

5.1.2 Swimming Operators

For fish, swimming is related to the important individual and collective behaviours such as feeding, breeding, and escaping from predators, moving to more livable regions of the aquarium or, simply being gregarious. In FSS, the causes of swimming are grouped into three classes: (i) individual, (ii) collective-instinct and (iii) collective volition. Accordingly, three operators are proposed to model different types of movements of fish: Individual movement operator, instinctive movement operator and volitive movement operator. The specific implementations of these three swimming operators are described as follows:

i) Individual movement operator

Individual movement occurs for each fish at every cycle of the FSS algorithm. The swim direction is randomly chosen. After each random movement, the fish assesses whether the food

density there seems to be better than at its current location. If this is not the case, the individual movement of the fish does not occur. Soon after each individual movement, feeding operator is implemented. For this movement, we define a parameter "*step_ind*" calle individual step to control the movement step size. Each fish moves if the new position has more food than the previous position. In the implementation, the movement step size for each individual is generated by multiplying *step_ind* by a random number generated by a uniform distribution in the interval [-1,1], as shown in (5.2):

$$d_j^i(t+1)=d_j^i(t)+\text{rand}(-1,1) * step\_ind \quad \text{.....(5.2)}$$

Where $d_j^i(t+1)$ denotes the j th dimension of i the fish of after the move and $d_j^i(t)$ denotes the j th dimension of i the fish of before the move.

ii) Instinctive movement operator

After all fish have moved individually, a weighted average of individual movements based on the instantaneous success of all fish of the school is computed. This means that fish that had successful individual movements influence the resulting direction of movement more than the unsuccessful ones. When the overall direction is computed, each fish is repositioned. This movement is based on the fitness evaluation enhancement achieved, as shown in (5.3).

$$X_{t+1}^i=X_t^i+\frac{\sum_{i=1}^N \Delta_{indi}\{f(X_{t+1}^i)-f(X_t^i)\}}{\sum_{i=1}^N \{f(X_{t+1}^i)-f(X_t^i)\}} \quad \text{..... (5.3)}$$

where Δ_{indi} is the displacement of the fish i due to the individual movement in the FSS cycle.

iii) Volitive movement operator

This movement is devised as an overall success/failure evaluation based on the incremental weight variation of the whole fish school. In other words, this movement is implemented according to the overall performance of the fish school. The rationale is as follows: if the fish school is putting on weight (meaning the search has been successful), the radius of the school should contract; if not, it should inflate. This operator is deemed to help greatly in enhancing the exploration abilities in FSS. The fish-school inflation or contraction is applied as a small step drift to every fish position with regard to the school's barycenter. The fish-school's barycenter is obtained by considering all fish positions and their weights, as shown in (5.4).Collective-volitive movement will be inwards or outwards (relative to the the barycenter), according to whether the previously recorded overall weight of the school has increased or decreased in relation to the new overall weight observed at the end of the current FSS cycle. The implementation of volitive movement is shown in (5.5) and (5.6).

$$\text{Rarycenter(t)}=\frac{\sum_{i=1}^{N} X_t^i * W_t^i}{\sum_{i=1}^{N} W_t^i} \qquad (5.4)$$

For this movement, we also define a parameter called volitive step (*step_vol*). We evaluate the new position as in (5.5) if the overall weight of the school increases in the FSS cycle; if the overall weight decreases, we use (5.6).

$$X_{t+1}^i = X_t^i(t) - step\_vol * \text{rand}(0,1) * [X_t^i - \text{bari(t)}] \qquad (5.5)$$

$$X_{t+1}^i = X_t^i(t) + step\_vol * \text{rand}(0,1) * [X_t^i - \text{bari(t)}] \qquad (5.6)$$

Algorithm 5.1 Fish School Search

Initialize fish in the swarm

While maximum iterations or stop criteria is not attained do

for each fish *i* in the swarm do

a. update position applying the individual operator (5.2)

b. apply feeding operator

update fish weight according to (5.1)

c. apply collective-instinctive movement

update fish position according to (5.3)

d. apply collective-volitive movement

if overall weight of the school increases in the cycle

update fish position using (5.5)

elseif overall weight of the school decreases in the cycle

update fish position using (5.6)

end for

End while

5.2.3 Experimental Setting and Results

There are two major parameters need to be tuned in the experiments, which are step size for individual movement (st-i) and supersize for volitive movement (st-v). The results of preliminary tuning are in table 5.2. And the experimental setting and results on Hanoi network are given in

table 5.3.

Table 5.2.Preliminary Tuned Results for Two Major Parameters

Parameters	Tuned results
St-i	2
St-v	0.5

Table 5.3.Experimental Setting and Results. (Compared with standard PSO)

method	St-i	St-v	Pop size	NO. of iterations	NO. of runs	Best cost	Average cost
FSS	2	0.2	500	500	30	6.733×10^6	7.174×10^6
IPSO	--	--	500	500	30	6.369×10^6	7.110×10^6

CHAPTER 6

6 MACHINE LEARNING ENHANCED SEARCH ALGORITHMS

In section 1.2, we have introduced how Machine Learning techniques are used to improve Evolutionary Algorithms. In this thesis research, one of the major tasks is to further improve the swarm intelligence algorithms selected in Chapter 5 using different machine learning techniques and test them on benchmark networks.

6.1 Preserve Population Diversity

6.1.1 Population Diversity and Premature Convergence

Classical PSO usually suffers from premature convergence especially when solving complex multi-modal-search problems [39]. When premature convergence is present, the search process is apt to be trapped in local optima. This could get detrimental when the problem has high dimensionalities. Parameter tuning is a conventional way to improve PSO performance as well as address parameter convergence yet fast diversity losing is still a challenge.

Population diversity measures how diverse the population is and it is used as a detector of premature convergence. Many researchers have proposed their strategies to control the diversity

of the population so that premature convergence could be prevented. Most recent efforts to tackle this problem focus on the following three approaches below:

1) Topology control. Population topology has a significant effect on the performance of PSO. Different population topologies, such as circle, wheels and stars, may have different influence on the performance of the search strategy on different problems [43]. An appropriate topology could help solve the premature convergence problem. Kennedy and R. Mendes [44] proposed a social-network topology that mimics the different communication structure of social networks. A more recent study [45] indicates that PSO algorithms with a ring topology are able to locate multiple global or local optima. The rationale behind the strategy is that divided population could help preserve the diversity of the whole population to some extent. However, an obvious drawback is that a large population size is usually required to ensure the performance of different topologies.

2) Randomness injection. This is a way to preserve the diversity by arbitrarily introducing a certain level of randomness into the population when the diversity of population fades. Krink et al. proposed a method to inject diversity into the population by introducing a concept of "particle collision"[46]. In this model, there is repulsive force between two particles which are too close to each other. An attractive and repulsive PSO (ARPSO) is proposed by Monson and Seppi[47], which divides the optimization into two different phases: an attraction phase when individuals are attractive to each other by sharing the information of personal best position and global best position and a repulsion phase when individuals repel by its personal best position and best known global position. The repulsion phase is implemented when the diversity is below a threshold. The alternation of attraction and repulsion phase preserves the diversity while ensure

the convergence does occur. The randomness injection methods could help preserve the diversity but with the sacrifice of the fast convergence of PSO.

3)Hybridization. In order to introduce advantages of other search algorithms to PSO, researchers start to combine another algorithm with PSO to form a hybrid PSO. The first hybrid PSO was developed by Angeline by introducing a selection scheme[48]. A more recent trend is to hybridize a local search strategy with PSO. Extreme optimization (EO) is integrated into PSO framework to perform exploiting behaviour, which performs superiorly in preventing premature convergence. However, the overhead of fitness evaluation is increased largely due to the nature of high computationally expensive of EO. R. V. Kulkarni [49] proposes an Estimation of Distribution improved PSO in which the particle swarm is allowed to estimate the distribution of promising solution regions. During each iteration, each particle is updated by both the PSO equation and EDA sampling and the one with better fitness is used to replace the previous particle. The diversity is measured by the ratio between mean and the maximum of the fitness function of all particles in iteration. M. El-Abd [50] proposed another PSO improved by EDA that used a uniform distribution and a threshold to decide the choice between PSO equation and EDA sampling while updating each particle. The EDA in these two hybrid variants performs like a local search algorithm which might improve the overall performance of the PSO algorithm but might not preserve the diversity. Since it is possible for two different individuals to have the same fitness value, using fitness to measure the population diversity may be problematic, especially when the problem is highly dimensional. This is analogous to the situation in the nature where different genotype may represent the same phenotype (fitness).

In order to measure the population diversity quantitatively in this research, the following formula is adopted to calculate the diversity [51]:

$$\text{Diversity(S)} = \frac{1}{|S|.|L|} \cdot \sum_1^{|S|} \sqrt{\sum_1^N (p_{ij} - \bar{p_j})} \qquad (6.1)$$

Where S is the swarm; |S| is the population size; |L| is the length of longest the diagonal in the search space; N is the dimensionality of the problem; p_{ij} is the jth value of the ith particle and $\bar{p}_j$ is the average value of jth dimension. It is obvious that this diversity measure is independent of population size, the dimensionality of the problem as well as the search range of each dimension.

6.1.2 Use Shuffle process to preserve the population diversity of PSO

In order to preserve the population diversity and prevent the premature convergence of conventional PSO, a shuffle process is integrated into the iteration process of PSO algorithm when the population turns to be approaching the convergence. Specifically, "Shuffle" here means creating next generation using a random normal distribution. Standard deviations of diameters of each pipe in previous generation are used as threshold of the "shuffle" process: if the population diversity is below the threshold, the shuffle process will be activated. Mean value of the diameters of certain percent (is a tunable parameter of the algorithm) of top individuals in previous population is used as the mean of the normal distribution for generating the next generation. And the standard deviation used for shuffle is a tunable parameter. The flowchart of this proposed model is showing in following figure:

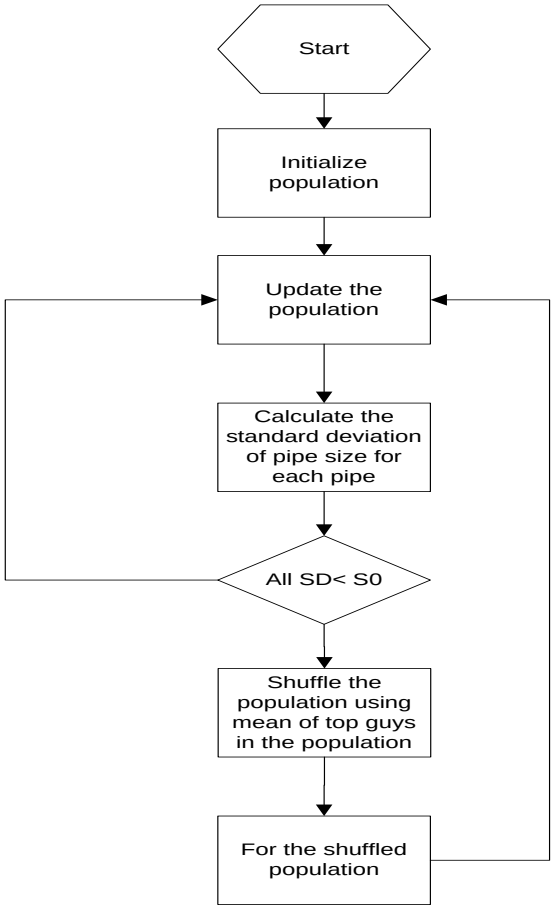


Figure 6.1.Flowchart of shuffled PSO

Apparently, there are 3 additional parameters need to be adjusted besides PSO parameters: Percentage of top guys (percentage), Standard deviation used for shuffle (std) and shuffle step size which means how often the shuffle process is used. Following are the results of fine tuning of the parameters in shuffle PSO (Hanoi network). In the experiment, we set the population size 500, and maxim number of generation is 500.

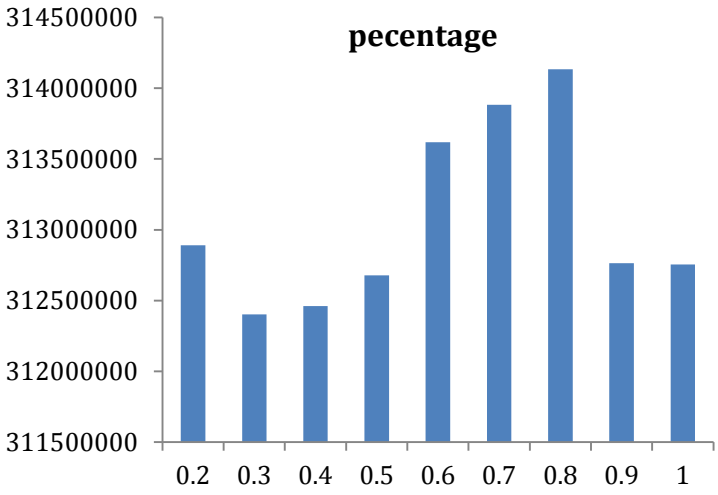


Figure 6.2.Performances over percentage of population used for shuffle

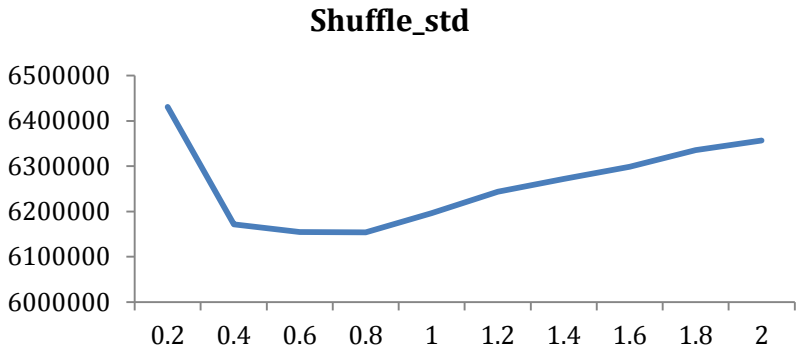


Figure 6.3.Performances over the standard deviation of shuffle process

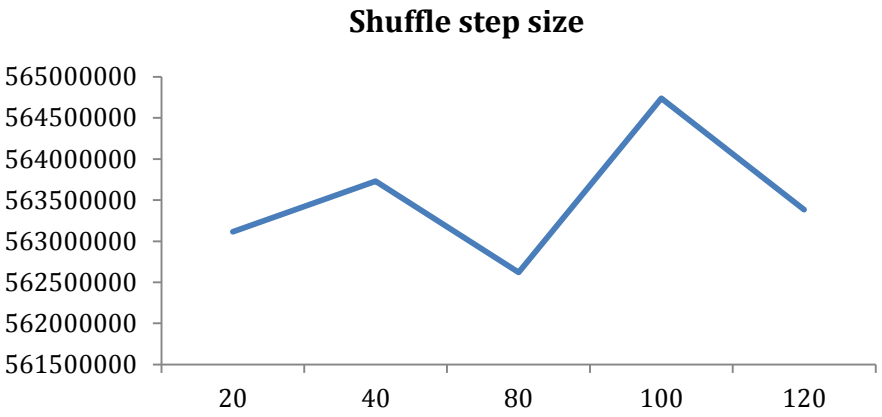


Figure 6.4.Performances over shuffle step size

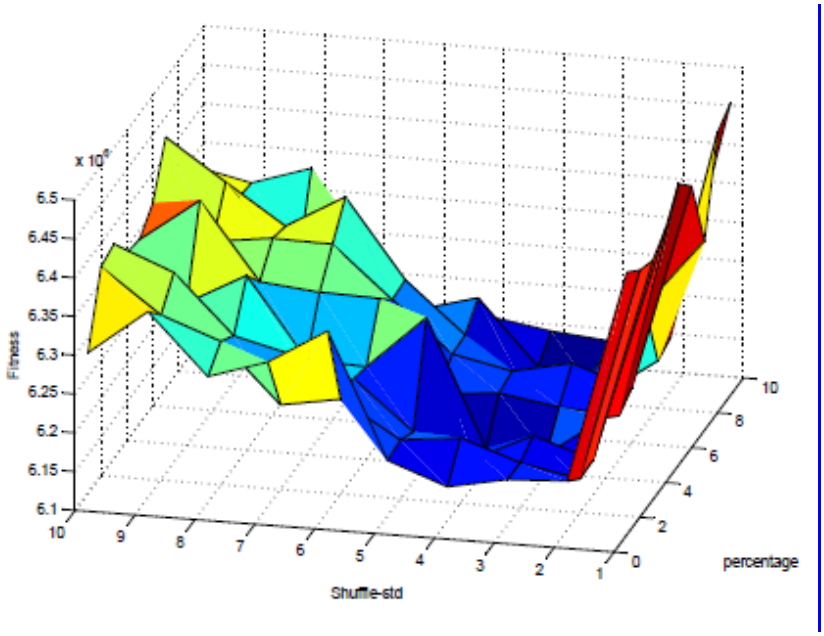


Figure 6.5.performances over shuffle-std and percentage (Hanoi network)

In the parameter setting experiments for the proposed shuffle-PSO, the three major parameters of PSO are kept the same all the time as the best parameter setting obtained from previous experiments. The tunable parameters are standard deviation of shuffle process (**shuffle-std**),

percentage of population used for shuffle (**percentage**) and shuffle step size (**INV**). Figure 6.3 help us identify the optimal value for Shuffle-std is around 0.8. The best value for **INV** is around 80, which is indicated by Figure 6.4. From Figure 6.5, we can conclude that the parameter shuffle-std dose affect the performance significantly, while the **percentage** doesn't have big impact on the performance. Finally, the optimal parameter setting is identified:

Shuffle-std =0.7

Percentage=0.3

INV=80

This proposed algorithm are tested on both medium size and large size benchmark examples, and then compared with standard PSO. The following table 6.1 and 6.2 are showing the comparison. And the fitness tracks on two benchmark examples are given in Figure 6.6 and Figure 6.7.

Table 6.1.Experimental Results on Hanoi Network(Shuffle-PSO)

	Population size	Number of iteration	C1 C2 w	percentage	Shuffle-std	Shuffle Step size	Number of runs	Average cost	Best cost
StandardPSO	500	500	2,2,0.8	--	--	--	--	$7.110x10^6$	6.3691x
Shuffle-PSO	500	500	2,2,0.8	80%	0.7	50	30	$6.299x10^6$	6.0811

Table 6.2.Experimental Rsults on Balerma Network(Shuffle-PSO)

	Population size	Number of iteration	C1 C2 w	percent	Shuffle-std	Shuffle Step size	Number of runs	Average cost	Best cost
StandardPS	1000	1000	2,2,0.8	--	--	--	--	$6.729x10^7$	6.299

O									
Shuffle-PSO	1000	1000	2,2,0.8	80%	0.7	50	30	4.2548×10^6	4.1256

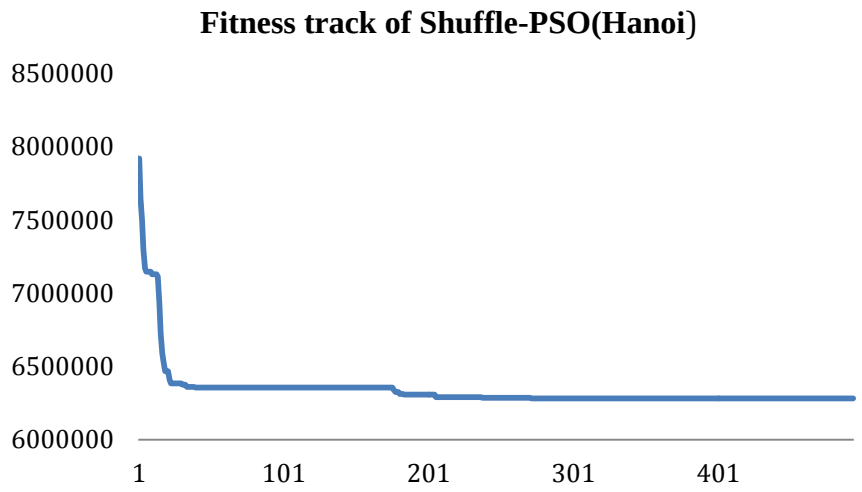


Figure 6.6.Fitness track of Shuffle-PSO (Hanoi Network)

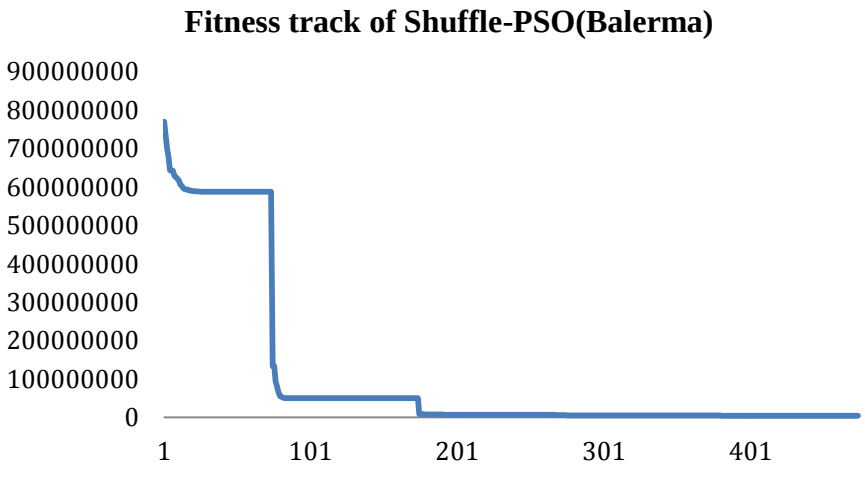


Figure 6.7.Fitness track of Shuffle-PSO (Balerma Network)

6.1.3 Use EDA to Preserve the Population Diversity of PSO

6.1.3.1 Particle Swarm Optimization

As introduced in Section 5.1, the updating equations (5.1) and (5.2) are given. For the sake of better understanding, besides the above particle-level updating equations, a population-level description using matrix is also given:

Let P_m^n(t) = {X_m^n(t), $\hat{X}_m^n$(t), $\Omega_m^n(t)$, $V_m^n(t)$ } be a configuration of the particle swarm in the current iteration. Where *n* is the population size and *m* is the dimensionality. *X*(t) represent the whole swarm,); $\hat{X}_m^n$(t) denotes all the personal best positions, $\Omega_m^n(t)$ is a matrix in which every row is the current global best particle; $V_m^n(t)$ is the velocity matrix of the current generations. Please note all above matrix are *n* x *m* matrix. So the updating process of the PSO could be simply described as follows:

X_m^n(t), $\hat{X}_m^n$(t), $\Omega_m^n(t)$, $V_m^n(t)$	=>	X_m^n(t+1) $\hat{X}_m^n$(t+1) $\Omega_m^n(t+1)$ $V_m^n(t+1)$

What is noteworthy is the relationship among X_m^n(t), $\hat{X}_m^n$(t), $\Omega_m^n(t)$. For each individual (a row in the matrix) in $\hat{X}_m^n$(t), it is updated if the corresponding individual in X_m^n(t) is better. And all the individuals in $\Omega_m^n(t)$ will be updated with the best individual in $\hat{X}_m^n$(t) if it is better. So the average fitness of all the individuals in $\Omega_m^n(t)$ is better than $\hat{X}_m^n$(t) and $\hat{X}_m^n$(t) is better than X_m^n(t). following figure 6.8 provides a general flowchart of using PSO on WDN optimization problem.

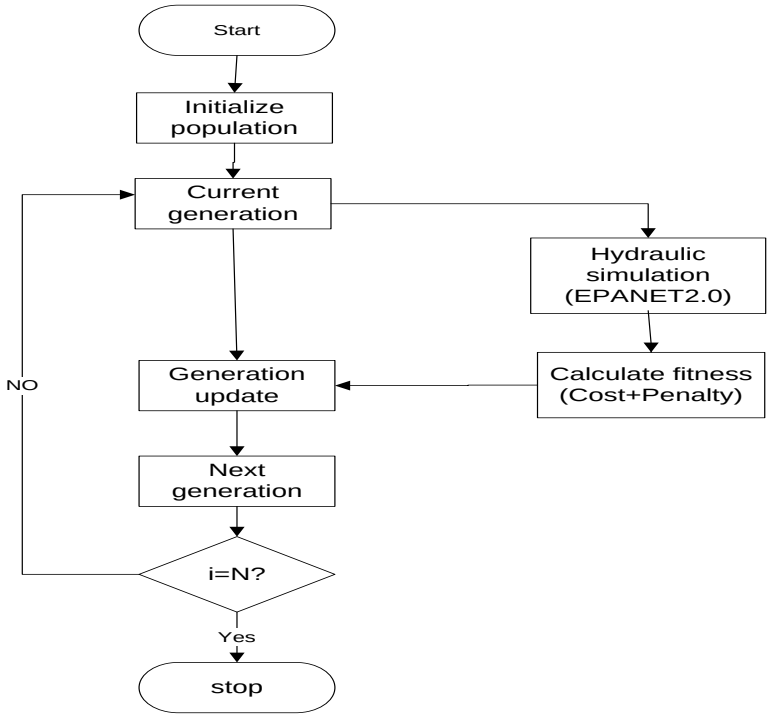


Figure 6.8.General flowchart of PSO on WDN optimization problem

6.1.3.2 Estimation Distribution Algorithm

Estimation of distribution algorithms (EDAs) are evolutionary algorithms that derived from the genetic algorithm (GA). But unlike GA, EDAs only have one updating operator. During the iteration process, EDAs estimate the probability distribution by using selected individuals to construct a probabilistic model. The constructed probabilistic model is then used to predict the promising region and generate the next new generation around that region. It is noteworthy that the probabilistic model in the EDAs is continuously updated. The best solution generated in each generation continually approaches the global optima as the probabilistic model approaches the unknown actual distribution. The typical structure of an EDA is shown in Algorithm 6.1 below:

Algorithm 6.1 Estimation distribution algorithm

1: P<= Initialize the population

2: Evaluate the initial population

3: While iter_number ≤ Max_iterations do

4: P_s<= Select the top s individuals from P

5: M<= estimate a new Model from P_s

6: P_n <= Sample n individuals from M

7: Evaluate P_n

8: P<= Select n individuals from P U P_n

9: iter_number =iter_number +1

10: end While

The key of the EDA algorithm is the way to estimate the probability distribution. A simple Gaussian distribution could be assumed for the distribution of pipe diameters when the individuals are approaching the global optima. In each iteration, a certain percentage of the top individuals in the population are used to estimate the mean and standard deviation of the distribution of each pipe. The new population is generated by sampling the estimated distribution. Figure 6.9 describes how the estimation and sampling work:

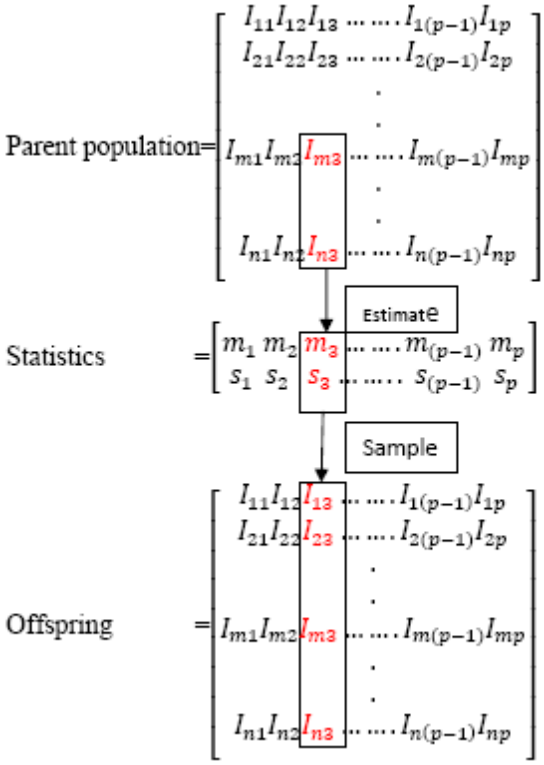


Figure 6.9.Flow chart of distribution estimation (sorted by descending order)

Where n is the population size; p is the dimensionality of the problem; m is the mean value of corresponding dimension; s is the standard deviation of the dimension. The above example shows the process of updating diameters of pipe3 in the offspring population by the Gaussian distribution calculated according to the top individuals in the parent population.

The key issue of applying EDA to improve PSO is to find the right sample for the construction of the probabilistic distribution in EDA process. A good sample could more accurately cover the most promising region of the search space and guide the search toward it, so the quality of the sample directly affect the accuracy of the probability distribution model which is built on it. In the previous work of hybrid EDA and PSO, the better half of personal best position ($\hat{X}_m^n$(t)) and

the current swarm (X_m^n(t)) are the most commonly used sample for EDA (see table 6.3), however a more appropriate sample could be used, which is called "historical good positions". We use H_m^n(t) to denote it. H_m^n(t) records the top best individuals that all the swarm has ever travelled rather than the best positions each particle has ever travelled. Why H_m^n(t) is better than $\hat{X}_m^n$(t)? Here is the explanation: we assume the following configuration of the particle swarm at two consecutive generations, which are shown in table 6.4. We assume the population size is 5(n=5) and the fitness value of each individual in different matrix are given (assume it is a maximization problem). As we can see, in generation t+1, the personal best position of the second particle is updated from 5 to 6, and the particle with fitness 5 is disregarded. But the particle with fitness 5 is actually a very good one which is successfully kept in H_m^n(t+1). So if we use the top 40% of the $\hat{X}_m^n$(t+1) as sample, it would be the particles with fitness value 6 and 4, but if we use top 40% of H_m^n(t+1) as the sample , it would be the particles with fitness value 6 and 5. Therefore, the top individuals in historical best positions are better sample positions for EDA. In the following sections, an improved hybrid algorithm of PSO and EDA is proposed using the historical best positions and it is compared with previous work.

Table 6.3.Sample Used in The Literature

Literature	Sample used for EDA
Y.Zhou and J. Jin[52]	Better half of X_m^n(t)
Mudassar Iqbal[53]	Top ones of $\hat{X}_m^n$(t)
R. V. Kulkarni[54]	Better half of $\hat{X}_m^n$(t)
M. El-Abd[55]	Better half of $\hat{X}_m^n$(t)
M. El-Abd [56]	Top ones of X_m^n(t)

Hongcheng Liu[57]	Better half of $\hat{X}_m^n$(t)
Chang Wok Ahn [58]	Top ones of local $\hat{X}_m^n$(t)

Table 6.4.Explanation of Historical Best Positions

	Generation t				Generation t+1			
	X_m^n(t)	$\hat{X}_m^n$(t)	$\Omega_m^n(t)$	H_m^n(t)	X_m^n(t+1)	$\hat{X}_m^n$(t+1)	$\Omega_m^n(t+1)$	H_m^n(t+1)
Fitness	2	3	6	6	4	3	6	6
	5	**6**	**6**	**3**	**4**	**6**	**6**	**5**
	1	2	6	2	3	2	6	4
	3	1	6	2	3	3	6	3
	4	2	6	1	5	4	6	2

6.1.3.3 Population Diversity

As analysed in the introduction part, an accepted hypothesis is that maintenance of high diversity is crucial for preventing premature convergence in multi-model optimization. Before discussing how to preserve the population diversity, we need to first define how to measure population diversity. There two different types of diversity by measuring from genotype and phenotype space. The diversity of phenotype space is indicated by the distribution of fitness values of

individuals in the population. It is calculated by the following equation:

$$\text{Diversity } (S) = \sqrt{\frac{1}{N}\sum_1^N (f_i - \bar{f})^2} \quad (6.1)$$

Most of the time the diversity of phenotype is not enough to indicate the real population diversity when there are too many different individuals that have the same fitness values, which is more likely to be present in the problem with high dimensionality. Hence, we choose genotype diversity in the research of this paper. In order to measure the population diversity of genotype space quantitatively, the following formula is adopted to calculate the diversity [51]:

$$\text{Diversity } (S) = \frac{1}{|S|.|L|}. \sum_1^{|S|} \sqrt{\sum_1^N (p_{ij} - \bar{p_j})} \quad (6.2)$$

Where S is the swarm; |S| is the population size; |L| is the length of longest the diagonal in the search space; N is the dimensionality of the problem; p_{ij} is the jth value of the ith particle and $\bar{p}_j$ is the average value of jth dimension. It is obvious that this diversity measure is independent of population size, the dimensionality of the problem as well as the search range of each dimension.

6.1.3.4. Convergence Process

How convergence happens in PSO? Before we know how to address premature convergence problem of PSO, we need to know the convergence mechanism of PSO first. According to equation (5.1) and (5.2), in a classical PSO updating process, the position of next step is decided by the velocity of next step (see equation5.2). And the velocity update of particles consists of three parts: the first is the inertia of particles; the second is cognitive acceleration which represents the particle's own experience; and the third is social acceleration which represents the social interactions between particles. So when the latter two parts are close to 0, the velocity will be close to 0. That is to say when most of the particles are approaching their personal best

positions and all the personal best positions are very close to each other, the convergence happens. At the moment, the population tends to lose its diversity and a particle could leave the place only if the inertia weight and its current velocity is not equal to 0. If the current position coincides with the global best position by chance, this particle stops improve, and when all particles becomes the global best position, the population completely converge. However, if this global best position is not the global optimum of the search space, premature convergence occurs.

6.1.3.5. Improved Sequential hybridization of PSO and EDA

Inspired by the above analysis of premature convergence, a potential way to prevent the premature convergence could be "boosting the personal best position ($\hat{X}_m^n(t)$)", which means to improve the personal best positions $\hat{X}_m^n(t)$ when the particles are very close to them by borrowing the strength from the outside, such as EDA. This could also explain the reason why hybridizing with EDA could improve the performance of PSO. In previous version of hybrids of PSO and EDA, EDA is used to estimate better positions and replace the current personal best positions with the better ones, so that the current best positions are boosted and the convergence is therefore delayed. As we discussed in previous section, however, in most of the previous version of hybrid PSO and EDA, top good individuals of personal best positions or just the current swarm are used to construct the estimation model in EDA. Hence, in order to further improve the "boosting" ability of EDA, we adopt the historical best positions as the sample positions for building the probabilistic model in EDA. This will result in more space for fitness improvement when the search process is close to convergence. Please note the EDA process is not started at the first generation, since at the very early stage of PSO search, all the particles are not in very

good positions, so the estimation model built on these positions should not be useful. Therefore, EDA process is started only M_s(tuneable) generations after PSO started. After EDA process starts, it is not implement at each iteration but only activated every certain number of iterations (M_f), which is also tuneable. The Pseudo-code of the Improved Sequential PSO-EDA (ISEDPSO) algorithm is illustrated below:

Algorithm 6.2 ISEDPSO

1: Initialization for PSO: set the initial positions: X_m^n(t=0), initial velocities $V_m^n(t=0)$, personal best positions $\hat{X}_m^n$(t=0) , global best position $\Omega_m^n(t=0)$ and historical best H_m^n(t=0). (n is population size, m is dimensionality).

2: While t ≤ Max_iterations && t > M_s do

4: Update the X_m^n(t), V_m^n(t), $\hat{X}_m^n$(t), $\Omega_m^n(t)$ according to equation (1) and (2).

5: If MOD(t, M_f)==0

6 Rank the mixture of $\hat{X}_m^n$(t) and H_m^n(t-1) according to their finesses (ascending).

7: Take 50% of the top individuals in the mixture as H_m^n(t) as sample for EDA.

8: Estimate the probabilistic distribution model using H_m^n(t).

9: Generate n new individuals $T_m^n(t)$ using the estimated distribution model and evaluate them.

10: Mixture $T_m^n(t)$ and H_m^n(t), rank the mixture according to fitness (ascending).

11: Take better half of the mixture as H_m^n(t).

12: Use H_m^n(t) to update $\hat{X}_m^n$(t) again.

13 End If.

14: t =t +1;

15: End While

In order to verify the advantage of this improved sequential hybridization of PSO and EDA, two other variants of this algorithm are also designed and will be tested and compared with ISEDPSO in the experiment phase. The only difference between the proposed algorithm and the two variant is the sample they used for EDA process. Specifically, SEDPSO-1 use better half of personal best positions and SEDPSO-2 adopts better half of swarm positions.

6.1.3.6. Parallel Combination of PSO and EDA (PEDPSO)

In previous version of hybridization of PSO with EDA that discussed in the literature review, although they have different specific implementation strategies, there is a common characteristic that the PSO updating process and EDA updating process are implemented sequentially. They either update the particles using PSO and EDA alternately, or select PSO and EDA updating process probabilistically for each particle. In our proposed ISEDPSO, PSO and EDA updating processes are implemented alternately for the whole population. Is there any other way to cooperate the PSO and EDA updating process? In this study, beside ISEDPSO, another novel parallel hybridization strategy for PSO and EDA is proposed to address the premature convergence problem. In the parallel hybrid algorithm, a standard PSO and a simple EDA process are combined in parallel rather than sequentially as depicted in Figure 6.10.

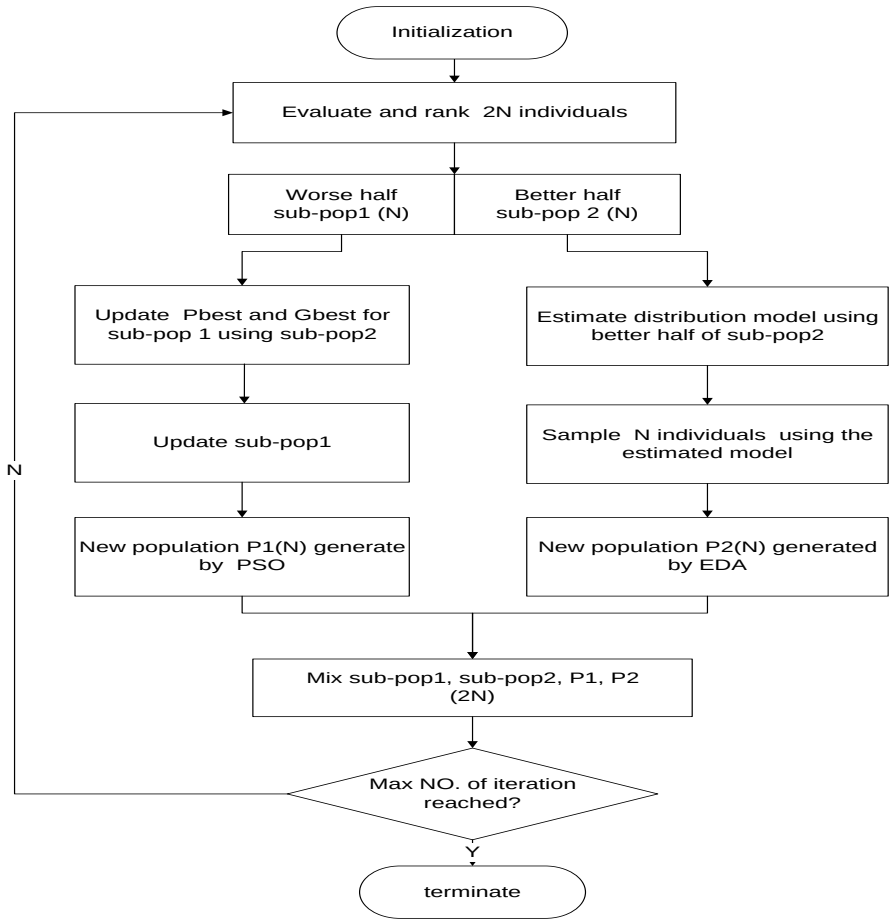


Figure 6.10.Flow chart of PEDPSO

The parallel algorithm starts from ranking and splitting the whole population into two sub populations: the better half is going to be updated by a typical EDA updating process, the worse half is updated by a typical PSO process. The reason behind this choice is that PSO is strong in improving very bad positions very fast (fast convergence ability), and EDA have to use a set of good positions to construct the correct probabilistic distribution model for estimation. At the end, both of the two new populations (P1 and P2) generated by PSO and EDA process, as well as the current population are mixed together and then better half of them is selected as the next new generation. There are two basic features and also the advantages for this novel hybrid strategy:

1). the population is actually the historical best positions throughout the search process, because at each iteration, we mix the current population and the newly generated population and then select the better half as next generation. This ensures that the search process always starts from very good positions the swarm has ever travelled. In terms of the EDA part, as we have discussed before, using historical best positions for constructing the distribution model is better than using personal best positions or good positions in the swarm. And 2) as we use the better half of the population as the personal best positions when updating the worse half of the population in PSO process, there is always space left for the swarm in PSO process to improve and the convergence process is therefore slowed down. This strategy is right similar to "boosting personal best positions" strategy used in ISEDPSO but it is more effective since at each iteration, before the population is completely convergent, the personal best position for each particle in PSO process is guaranteed to be better than the particle, while in previous sequential hybrids, it is not guaranteed. So this is the most distinctive part for this new hybrid method with respect to the previous sequential hybrids of PSO and EDA. In sum, this hybridization strategy is trying combine the advantage of PSO in fast convergence and the advantage of prediction of EDA together to control the population diversity and prevent premature convergence problem.

6.1.3.7 Parameter Settings

In the experiment phase, we compare the proposed two algorithms (ISEDPSO and PEDPSO) with the standard integer PSO and two other different versions of ISEDPSO in which the personal best positions and swarm are used by EDA process. We not only compare the best results they can achieve but also compare the diversity control ability of each algorithm. And then the two proposed algorithm in this paper will be compared with other algorithms for WDN

optimization from the literature on two benchmark networks.

In order to make the results more comparable, PSO parameters are set uniformly. Both C1 and c2 are set to 2; inertia weight is 0.8, V_{max} is 50%, which denotes the absolute value of the boundary when updating velocity of each particle. Each algorithm runs for 30 times on Hanoi network and Balerma network. The population size is set as 500 and the maximum number of generations is 2500. For the number of generations before EDA process (M_s) starts and the frequency of implementing EDA in ISEDPSO and its variants (M_f), preliminary experiments are carried out and a proper parameter setting is identified: M_s=100, M_f=50.

6.1.3.8 Experiment Results and Analysis

The experimental results are shown in Table 6.5. As we can see, for Hanoi network, both of the proposed algorithms achieve the known best results in the literature. For Balerma network, only ISEDPSO and PEDPSO could achieve best cost below 2M€. IPSO and ISEDPSO-2 performs much worse than the others on both of the networks.

Table 6.5.Experimental Results.

Algorithm	Hanoi Network(M$)		Balerma Network(M€)	
	Best cost	Average cost	Best cost	Average cost
IPSO	$6.369x10^6$	$7.110x10^6$	$6.299x10^6$	$6.729x10^7$
ISEDPSO-2	$6.412x10^6$	$7.321x10^6$	$6.715x10^6$	$7.062x10^6$
ISEDPSO-1	$6.081x10^6$	$6.210x10^6$	$2.021x10^6$	$2.043x10^6$
ISEDPSO	$6.081x10^6$	$6.102x10^6$	$1.932x10^6$	$2.026x10^6$
PEDPSO	$6.081x10^6$	$6.113x10^6$	$1.921x10^6$	$2.002x10^6$

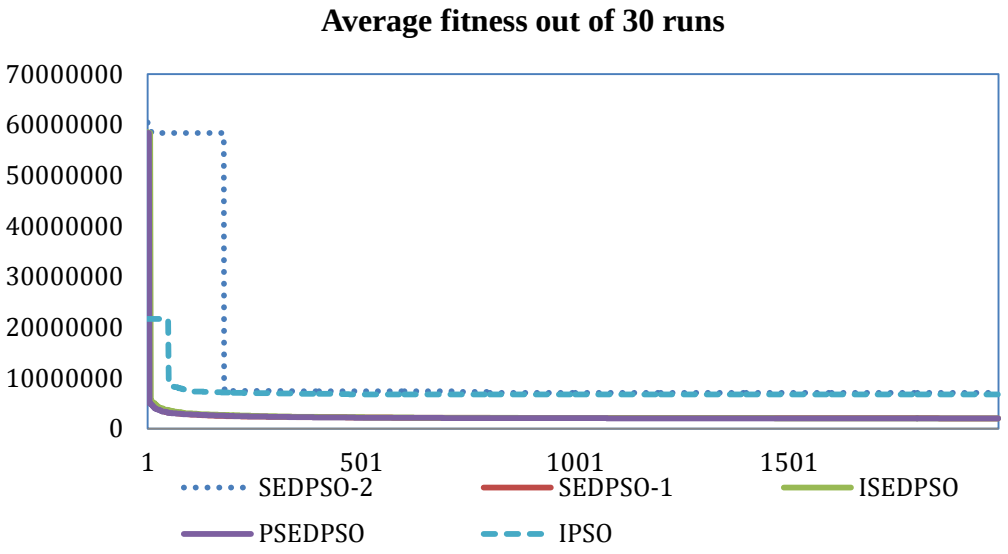


Figure 6.11.Fitness track of different algorithms (Balerma network)

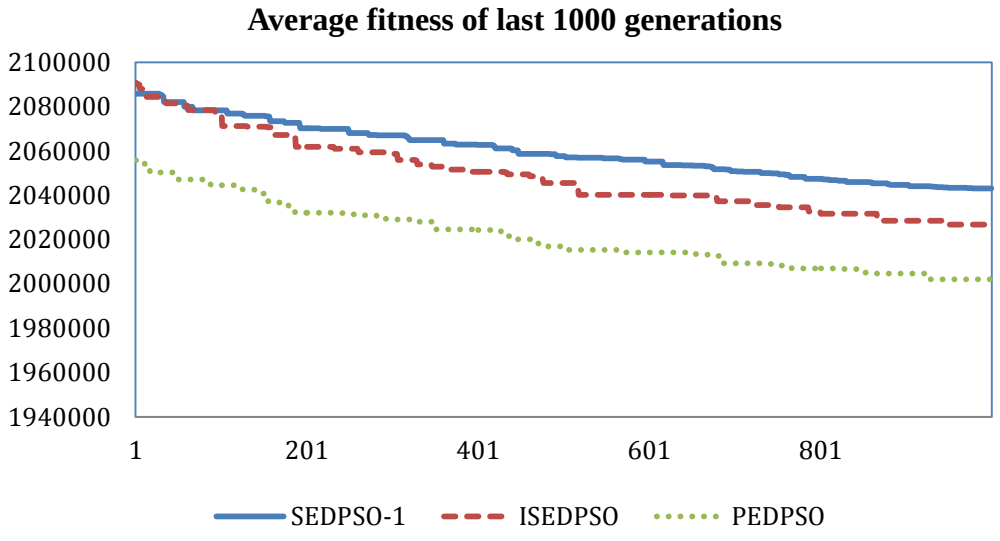


Figure 6.12.Fitness track of last 1000 generations (Balerma network)

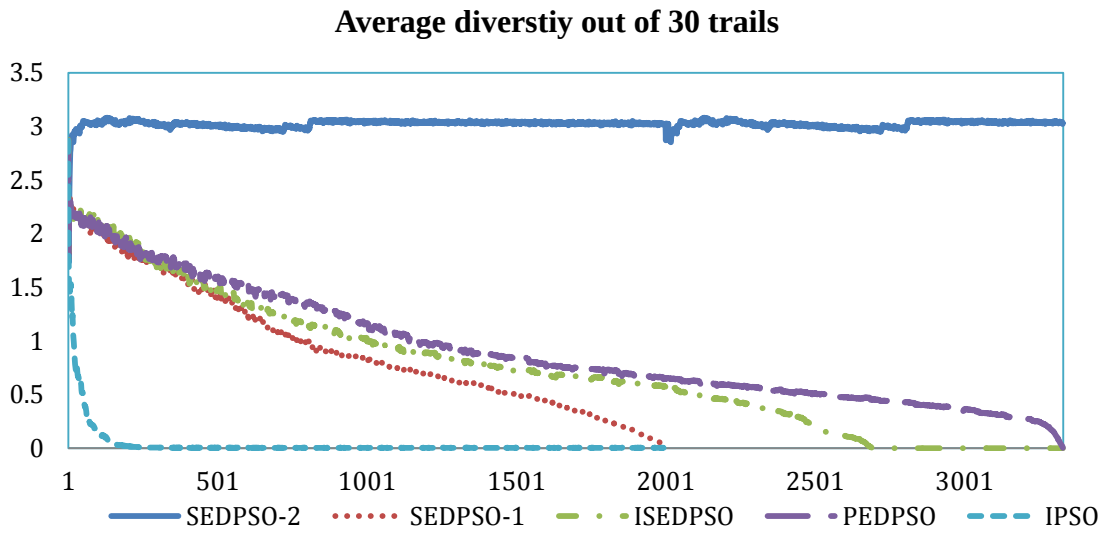


Figure 6.13.Diversity tracks of different algorithms (Balerma network)

In order to find out the relationship between the search performance and diversity control ability of the algorithms, the change of average fitness and diversity of the population in the experiment on Balerma network are both tracked and recorded. Figure 6.11 and Figure 6.12 show the fitness track of the algorithms. It is obvious that IPSO and SEDPSO-2 converge very fast and fail to achieve very good results. The other three algorithms including the two novel algorithms proposed in this paper could achieve much better results and the track of the last 1000 generations can be observed in Figure 6.12. It shows that the two proposed algorithms outperform the others and PEDPSO is one with the best performance.

What is noteworthy is that these results could be explained by the diversity track of the search process which is shown in Figure 6.13. The population diversity is calculated according to

equation (6). As we can see, conventional integer PSO losses population diversity very quickly and therefore results in premature convergence. On the contrary, SEDPSO-2 could maintain very high diversity throughout the search process, which causes an almost random search. So in both of these two extreme situations, the search algorithms could not achieve good results. The other three algorithms successfully control the population diversity gently and let it decrease more smoothly. As we can see in Figure 6.13, the diversity of SEDPSO-1 decrease slowly but it is close to 0 after only 2000 iterations. While the two proposed algorithms (ISEDPSO and PEDPSO) always maintain higher diversity through the search process and approach 0 at around 500 and 1000 iterations later.

From the analysis, we are able to make at least following four conclusions: 1) the diversity control ability of the search algorithm affect the overall performance of the algorithm. 2) A smooth decreasing trajectory of the population diversity throughout the search process could help prevent premature convergence. 3) Using historical best position in EDA process could really help PSO to better control the diversity and achieve better results. And 4) it is obvious that both of the proposed new hybrids of PSO and EDA effectively control the population diversity and prevent the premature convergence problem.

6.1.3.9 Compare with previous methods for WDN optimization

In order to justify the advantages of our proposed algorithms over previous methods, we compare the proposed algorithms with previous work extensively. We select several most recent related works which all adopt 10.6668 as Hazen-Williams roughness coefficient for hydraulic calculations on Hanoi network and Darcy–Weisbach roughness coefficient 0.0025 for Balerma network. The selected work also includes all the previous attempts of applying PSO to WDN

optimization problem. Please note some of the previous work did not test their method on the larger Balerma network.

Table 6.6(a) shows that both ISEDPSO and PEDPSO can achieve the best reported cost for Hanoi network. Although the number of evaluation calls of both proposed algorithms (17,600 and 23,400 respectively) are a little higher than the best of the other models(16,600), the proposed two novel algorithms perform very well in terms of reliability: The successful rates for ISEDPSO and PEDPSO are 93% and 90% respectively, only one previous work achieved better successful rate that is over 90% . Since Hanoi network is relatively smaller and is not difficult for most of algorithm to identify the known best cost (6.0811 price units), most of the recent methods could achieve this result. So no one was able to find a better result in including our two proposed algorithms. While on Balerma network, the situation is completely different, so far no commonly recognized best result has ever been identified.

Table 6.6(a). Comparison of The Best Cost Achieved by Different Algorithms (Hanoi)

Algorithms	Min cost	Average	Success rate	Min # of evaluations
GA([32] 2006)	6.173	n/a	n/a	26,457
ACO([57].2006)	6.134	n/a	n/a	35,433
HS([58], 2006)	6.081	n/a	1/81	27,721
PSHS([59],2009)	6.081	n/a	1/81	17,980
GHEST([39], 2010)	6.081	n/a	6/10	16,600
PSO ([35],2008)	6.133	n/a	3/100	n/a

HD-DDS([60], 2009)	6.081	6.252	8/100	5000,000
SAPSO([38],2010)	6.081	n/a	1/100	n/a
NLP-DE([61],2011)	6.081	n/a	98/100	48,724
ISEDPSO(this work)	**6.081**	**6.102**	**28/30**	**17,600**
PEDPSO(this work)	**6.081**	**6.103**	**27/30**	**23,400**

(In all algorithms, Hazen-Williams roughness coefficient for hydraulic calculations is 10.6668. Price unit: M$)

Table 6.6.(b).Comparison of the Best Cost Achieved by Different Algorithms (Balerma)

Algorithms	Min cost	Average	Success rate	Min # of evaluations
GA([32], 2006)	2.302,000	n/a	n/a	10,000,000
HS([58], 2006b)	2.018,000	n/a	1/81	10,000,000
HD-DDS([60], 2009)	1.940,923	2.165,000	n/a	30,000,000
GHEST([39], 2010)	2.002,387	n/a	1/10	290,500
NLP-DE([61], 2011)	1.923,000	1.927,000	n/a	1427,850
ISEDPSO(this work)	**1.971,460**	**2.026,672**	**16/30**	**89,700**
PEDPSO(this work)	**1.933,407**	**2.002,231**	**17/30**	**217,400**

(In all algorithms, Darcy–Weisbach roughness coefficient for hydraulic calculations is 0.0025, Price unit: M€)

Table 6.7. Comparison with the same number of evaluations (Balerma)

Algorithms	Min cost	NO. of evaluations
GA([36]., 2008)	3,738	45,400
SA([36], 2008)	3.476	45,400
MSATS([36], 2008)	3.298	45,400
HS([58], 2006)	2.601	45,400

PSHS([59], 2009)	2.633	45,400
MA([37], 2010)	3.120	45,400
GHEST([39], 2010)	2.178	45,400
ISEDPSO(this work)	**2.1083**	**45,400**
PEDPSO(this work)	**2.3781**	**45,400**

(In all algorithms, Darcy–Weisbach roughness coefficient for hydraulic calculations is 0.0025, Price unit: M€)

Table 6.6(b) shows the optimization results of Balerma network. Both ISEDPSO and SEDPSO are tested 30 times on this larger size network and both of them can achieve a very good cost with much less computation effort, compared to the literature. Specifically, ISEDPSO successfully achieved a new record of the best result, which is 1.921 unit price, using only 201,400 fitness evaluations. PEDPSO could also achieve a very good result (1.933 price unit) using much less fitness calls (217,400) comparing to the most recent work [59]. So, both of the proposed algorithms in this work outperform all the previous work in terms of the efficiency. And the high efficiency on this larger network indicates they have great potential ability on even larger real-world networks.

In order to further reveal the potential of our proposed methods, the proposed algorithms are compared according to the best results each algorithm could achieve with the same evaluation calls. Since only a limited number of previous work provides such data, we only includes several previous methods in this comparison. Table 6.7 shows that SEDPSO achieved the lowest cost within the required evaluation calls(45,400), while PEDPSO cannot obtain lower cost but the result still can rank the second in the available results in the literature. So we can say that these two algorithms still maintain certain level of fast convergence ability at the early stay of search which is the inherent advantage of PSO, although we already use EDA to control the population

diversity and prevent the fast convergence process.

We have to mention the most recent method which achieved minimal cost on Balerma network [59]. Comparing to this best method from the literature, we believe our method has better generality and easy to implement and extend, especially when we need include other considerations in the optimization, such as the reliability of the water network. In such situations, the method in [59] will have to adapt large part of their methods, while our method only need to convert the new consideration into numerical evaluation and add it into the calculation of fitness values. Moreover, when we apply our method into a new network, the only thing we need to change is the length of each individual, but in [59], all the implementation of NLP and shortest-distance tree search algorithm will have to be modified accordingly.

6.1.4 Clustering Analysis enhanced Fish School Search

Clustering Analysis (CA) has been used to improve many other swarm intelligence algorithms, such as PSO. In most cases, CA is used to split the whole population into smaller sub populations and each of the sub population evolves separately so that the diversity of the entire population could be preserved. It is also very suitable for further enhance the search ability of Fish School Search due to the following two reasons: 1) from the observation of natural behavior of fish schools, the populations size of the fish school is dynamic and small size schools are more reliable. And 2) in the instinctive movement operator of FSS, all the movements of the fishes in next iteration, are decided by the good movements of previous iteration. If the fishes are located around different local optima(or landscape), it is not reasonable to move the fish according to the

good move direction of the fishes around other local optima.

Upon aforementioned analysis, a dynamic seeded Clustering Analysis is used to improve the Fish School Search. The basic idea of predefined radius R (tunable) to split the entire school into small sub schools, when a fish is within the distance of R_i to the seeding fish of one school, then the particle will be absorbed into that sub school, otherwise if there no sub school could absorb that fish, it will be a seed for a new fish school. Algorithm 6.1 provides the details for this dynamic clustering procedure. After the entire fish school is split into small sub schools, each sub school is then updated according to a complete FSS updating process(Algorithms 5.1), therefore the number of the sub schools are keep changing throughout the search process. This new variant of FSS is named as "Clustering Analysis enhanced Fish School Search (CAFSS)" The details of this new algorithm could be found in the following algorithm 6.2:

Algorithm 6.1 Dynamic clustering peocedure

1: Sort the population (P) according to their fitness (ascending). (Minimization problem)

2: While i < population size do

if P_i is not assigned into any sub swam

While i+1<= j< population size do

If P_j is not been assigned to any sub swarm

If Distance(P_i, P_j)<= R

Assign P_j into the sub swarm which is seeded by P_i

End If

End If

j=j+1;

End While

else if

---

i=i+1;

End while

3. Output all the seeds of each sub swarm and membership of each fish.

Algorithm 6.2 CAFSS

1: Initialization of population: set the initial population (P), populations size(N) and Max iterations.

2: While iter_number ≤ Max_iterations do

Update membership of each fish according to Algorithm 6.1.

For each sub fish school, update all the fishes according to Algorithm 5.1.

3: iter_number =iter_number +1

4: end While

6.1.4.1 Experimental Setting and Results

The proposed algorithm is tested on Hanoi network and compared with original FSS and standard PSO. There is one important parameter of this algorithm: the clustering radius R. So the first step of the application of this algorithm is to find a good value for R. As we know the possible biggest distance between any two points in the search space could be identified if we use Euclidian distance and we also know the value range of each dimension. Specifically, for Hanoi network, the biggest distance is D= $\sum_1^{34}(6-1)^2$. In the experiment, we tested all the following R: 1/10, 1/9. 1/8, 1/7, 1/5 and the track of number of clusters are given in Figure 6.14. As we can see that, when R is very small, such as less than D/8, the number of clusters almost not changes along the entire search process, which means it is hard to converge. On the contrary,

when R is very big, such as larger than D/7, the number of sub swarms is very small and also does not change too much along the search process, in such situation, the diversity of the population is not contained very well. Only when R=D/8, the number of clusters decrease smoothly from 300 to 100 hundred and then converges to 100(population size is 500). This is a very ideal situation for the balance between preserving the population diversity and keeping the fast convergence rate. Therefore, we use D/8 as the clustering distance and the other parameters are listed in the table 6.8. Experimental results are also shown in table 6.8.

Table 6.8.Experimental Setting and Results

Algorithms	St-i	St-v	Pop size	NO. of iterations	NO. of runs	Best cost	Average cost
FSS	2	0.2	500	500	30	6.733×10^6	7.174×10^6
Standard PSO	--	--	500	500	30	6.369×10^6	7.110×10^6
CAFSS	2	0.2	500	500	30	6.587×10^6	7.095×10^6

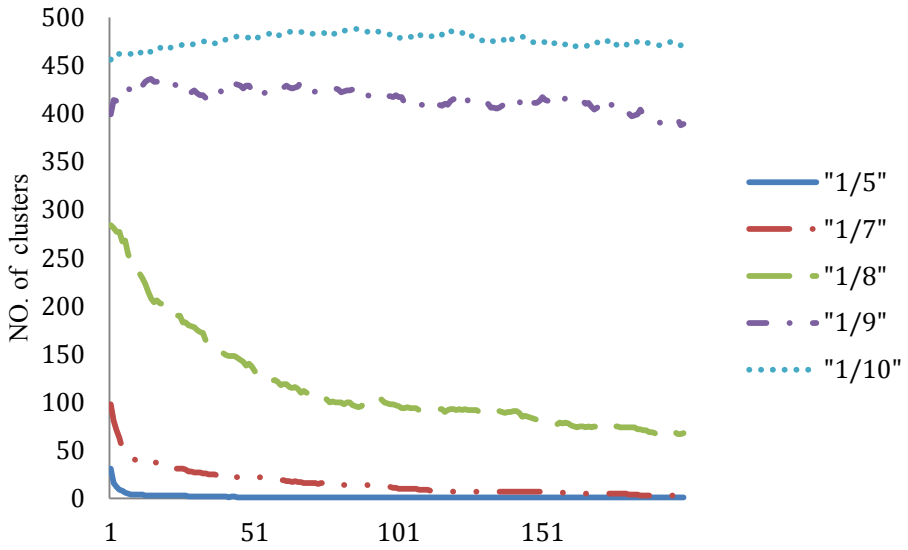


Figure 6.14.Track of the number of clusters at different R value (Hanoi)

Fitness track of one single run(Hanoi)

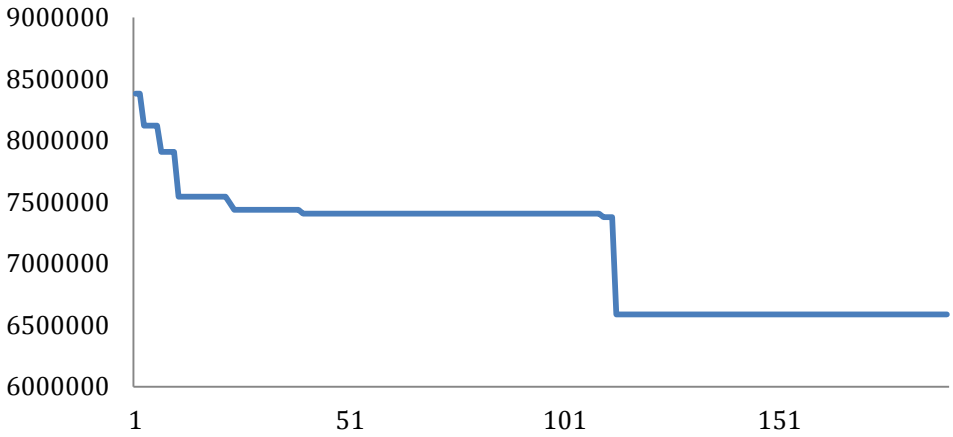


Figure 6.15.Fitness track of CAFSS on Hanoi network

6.2 Predict Promising Region in Search Space

6.2.1 Predicting Promising Region

Among the different ways of using ML to enhance EAs, Predicting Promising Region is a hot research area in the domain. During the search process of EAs, predicting promising positions or region according to previous data could help generate better population for the next generation and guide the search toward promising region where the optima is more like to be. Thus far, variants of ML techniques such as statistics and analysis methods have been used to predict promising regions in the search space and to guide the generation of new populations. Liang and Suganthan [60] proposed using a history learning strategy to create statistics on the historical data and to predict the promising moving direction of a particle when designing PSO algorithm. Zhang *et al*. [61] used a statistical method to design intelligent crossover operator for GP so as to predict a promising search region. Li and Tam [62] used the CA technique to cluster the individuals into different groups during the search process and preserved the best individual of each group to predict a promising search direction and to help generate a better population. CA techniques are also used in PSO to predict a promising leader for guiding the flying behavior of particles [63] and used in DE to generate promising new population according to the cluster centers [64].

Based on the above analysis of the literature, the basic idea of predicting the promising region is to first assume the good positions in the search space follow certain type of probability distribution and then estimate the distribution, at last, use the estimated distribution to predict the promising positions. Therefore, type of distribution estimated is the key issue and it may vary for different specific problems.

6.2.2 Particle Swarm Optimization

We select Particle Swarm Optimization here as the search algorithm. As we have introduced before, PSO is a population based optimization method inspired by the collective behavior of a bird flock.

Why we choose PSO here? There is an important property of PSO is that the best position each particle has ever traveled are all recorded and these particles could be regarded as the representative sample of good positions from the promising region of the search space, use follow figure 6.16 to illustrate: During the entire search process, personal best position for each particle keep updated. Whenever a particle been around the peak(a), the position will be recorded as the personal best position and at the later stage of the convergence, most of personal best positions will be around peak (a) although some of the them might be around other lower peaks. But the top individuals of personal best positions will most like all being round the highest peak. This analysis is the premise of the following experiments.

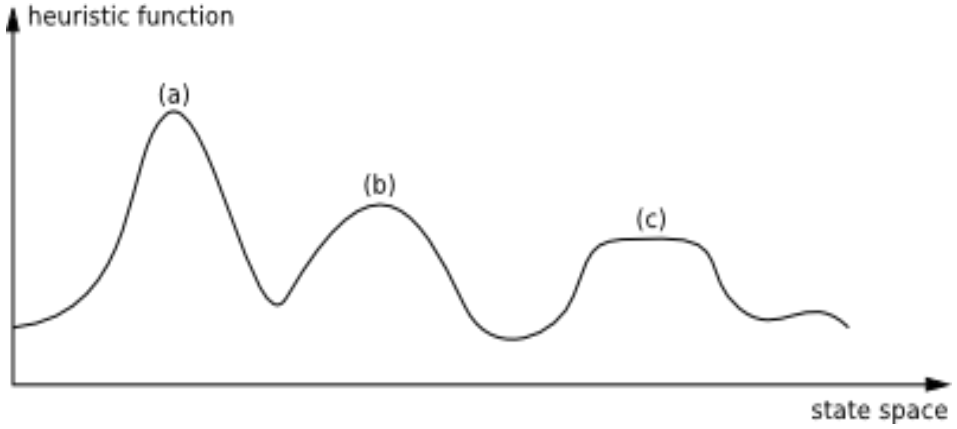


Figure 6.16.Illustration of personal best position

6.2.3 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. It can be given by following equation,

$$P(X|\lambda)=\sum_{i=1}^{M} \omega_i g(X|\mu_i, \Sigma_i), \quad (6.1)$$

where X is a D-dimensional continuous-valued data vector, ω_i, i=1…….,M, are the mixture weights, and $g(X|\mu_i, \Sigma_i)$, i=1,…….,M, are the component Gaussian densities. Each component density is a D-vitiate Gaussian function of the form,

$$g(X|\mu_i, \Sigma_i)=\frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(X-\mu_i)'\Sigma_i^{-1}(X-\mu_i)\}, \quad (6.2)$$

With mean vector μ_i and covariance matrixΣ_i. The mixture weights satisfy the constraint that: $\sum_{i=1}^{M} \omega_i = 1$.

The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation λ,

$$\lambda=\{\omega_i, \mu_i, \Sigma_i\} \quad i=1,.....,M. \quad (6.3)$$

Figure 6.17 shows a simple example of Gaussian Mixture Model when there are 3 Gaussian components.

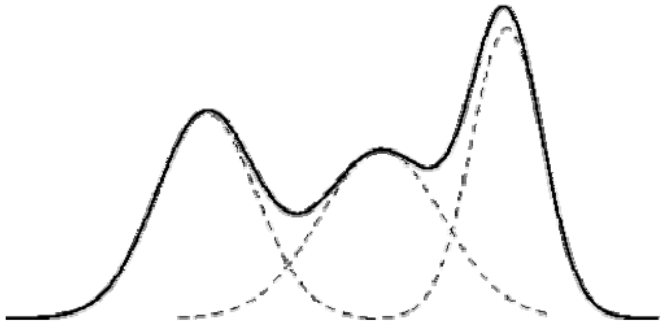


Figure 6.17.An example of Gaussian mixture with 3 components.

6.2.4 Expectation Maximization

Given training data (vectors) and a GMM configuration (number of components and probability for each component), we wish to estimate the parameters of the GMM, which best matches the distribution of the training data. There are several techniques available for estimating the parameters of a GMM [65]. By far the most popular and well-established method is maximum likelihood (ML) estimation.

The aim of ML estimation is to find the model parameters which maximize the likelihood of the GMM given the training data. For a sequence of T training vectors X=$\{X_1, \ldots \ldots X_T\}$, the GMM likelihood, assuming independence between the vectors, can be written as,

$$P(X|\lambda)=\prod_{t=1}^{T} p(X_t|\lambda) \qquad (6.4)$$

Unfortunately, this expression is a non-linear function of the parameters λ and direct maximization is not possible. However, ML parameter estimates can be obtained iteratively using expectation-maximization (EM) algorithm[66].

The basic idea of the EM algorithm is, beginning with an initial model λ, to estimate a new model λ, such that p(X|λ) >= p(X|$\bar{\lambda}$). The new model then becomes the initial model for the next iteration and the process is repeated until some convergence threshold is reached.

On each EM iteration, the following re-estimation formulas are used which guarantee a monotonic increase in the model's likelihood value,

Mixture Weights:

$$\overline{w_i}=\frac{1}{T}\sum_{t=1}^{T}\Pr(i|X_t,\lambda) \tag{6.5}$$

Means

$$\overline{\mu_i}=\frac{\sum_{t=1}^{T}\Pr(i|X_t,\lambda)X_t}{\sum_{t=1}^{T}\Pr(i|X_t,\lambda)} \tag{6.6}$$

Variances (diagonal covariance)

$$\overline{\sigma^2}_i=\frac{\sum_{t=1}^{T}\Pr(i|X_t,\lambda)x^2{}_t}{\sum_{t=1}^{T}\Pr(i|X_t,\lambda)}-\overline{\mu^2}_i \tag{6.7}$$

6.2.5 Assumptions for Distribution Model

Before we estimate the parameters of the assumed distribution, we need to make the assumptions for the distribution. We can assume the dimensions are independent with each other and each dimension follows a type of distribution; or we can also assume all the dimensions together follow a multivariate distribution. For this specific engineering problem, we can make following assumptions:

(1) Assume the diameters of different pipes are independent with each other. Good values of diameters of each pipe follow a mixture Univariate Gaussian distribution.

(2) Assume the diameters of different pipes are dependent with each other, which means that the good settings of diameters of all the pipes in the network follow a mixture of Multivariate normal distributions.

Based on above assumptions, two different algorithms are proposed in the next section.

6.2.6 Predict the Promising Region for PSO

According to the abovementioned assumptions, two different methods of predicting the promising region are proposed.

With either of the two assumptions, a mixture Gaussian model is assumed, and EM is used to estimate parameters for the mixture model, then the estimated model is used to re-sampling again. More specifically, during each iteration of PSO, the top 20% of the personal best

positions are used as the sample for estimation. After the re-sampling, the newly generated individuals are re-ranked with previous personal best positions and better half of them is adopted as the personal best positions for the next generation.

1) Univariate Gaussian Mixture

When assume the dimensions are independent with each other, it means the good diameter settings for each pipe in the network follow a mixture of Univariate Gaussian Distribution. We call the proposed algorithm based on this assumption "Univariate EM based Particle Swarm Optimization "(EMPSO-U).

Algorithm 6.3. EMPSO-U

1: Initialization for PSO: set the initial population(N): X_0 , initial velocities V_0 , $Pbest_0$, $gbest_0$ (c1=c2=2,w=0.5)

2: While iter_number ≤ Max_iterations do

4: Update the velocities V, positions X, $Pbest_i$, $gbest_i$ according to PSO strategy.

5: Rank the updated personal best positions according to their finesses.

6: Take 20% of the top individuals in $Pbest_{\mathrm{i}}$ (N) to estimate the distribution using EM for each pipe.

7: Regenerate N individuals according the estimated distribution.

8: Mix the newly generated with the previous personal best positions and take the better half as the Personal best positions for the next.

9 Update the $gbest_{\mathrm{i}}$.

10: iter_number =iter_number +1

11: end While

2) Multivariate *Gaussian Mixture*

When assume the dimensions are dependent with each other, it means the good diameter settings

for all the pipes in the network follow a mixture of Multivariate Gaussian Distribution. We call the proposed algorithm based on this assumption a "Multivariate EM based Estimation Particle Swarm Optimization "(EMPSO-M).

Algorithm 6.4. EMPSO-M

1: Initialization for PSO: set the initial population(N): X_0 , initial velocities V_0 , $Pbest_0$, $gbest_0$ (c1=c2=2,w=0.5)

2: While iter_number ≤ Max_iterations do

4: Update the velocities V, positions X, $Pbest_i$, $gbest_i$ according to PSO strategy.

5: Rank the updated personal best positions according to their finesses.

6: Take 20% of the top individuals in $Pbest_{\mathrm{i}}$ (N) to estimate the distribution using EM.

7: Regenerate N individuals according the estimated distribution.

8: Mix the newly generated with the previous personal best positions and take the better half as the Personal best positions ($Pbest_{\mathrm{i}+1}$)for the next.

9 Update the $gbest_{\mathrm{i}}$.

10: iter_number =iter_number +1

11: end While

6.2.7 Experimental Results and Analysis

Both of the proposed algorithms are tested on two benchmark networks. And also they are compared with a conventional integer PSO. The parameter setting for the algorithms are listed in Table 6.9.Parameter Setting for Experiments of EMPSO

Network	Assumption	Population size	Max # of generations	Percentage of pbest for sampling

Hanoi	Univariate	500	100	20%
	Multivariate	500	100	20%
	IPSO	500	100	NA
Balerma	Univariate	1000	200	20%
	Multivariate	1000	200	20%
	IPSO	1000	200	NA

In order to test the performance of the proposed search algorithms on different number of Gaussian components and identify the best number of Gaussian components, for each of above setting, the algorithms run 30 times on each of the following number of components: 1, 2, 3, 5, 8, 10, 20. Due to the fact that the convergence time for EM increase largely as the time of components increase, so larger number (>20) of components are not tested on this problem. From Figure 6.18 and 6.19, on both assumptions, the number of components does affect the performance of the PSO. And we know that when using Univariate assumption, 10 Gaussian mixture model is the best and when using Multivariate model, 2 Gaussian mixture is the best. With the best Assumed Gaussian model, the final experimental results on Hanoi network are recoded in Table 6.10.

Table 6.10.Experimental Results of EMPSO

Assumption	NO. of Gaussian components	Best cost	Average best cost	Standard deviation
Univariate	10	6.2361×10^6	6.2617×10^6	1.151×10^5
Multivariate	2	6.1138×10^6	6.1972×10^6	1.210×10^5

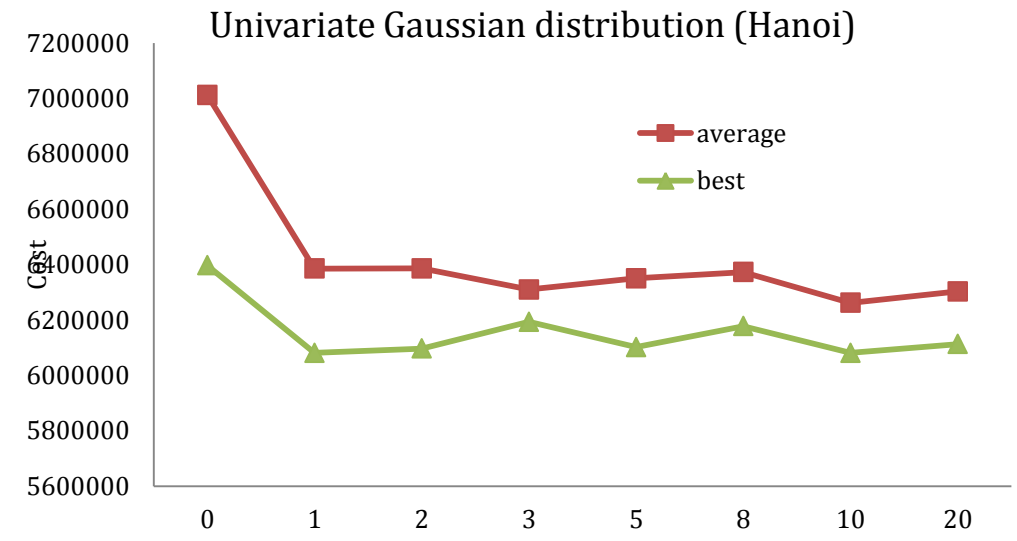


Figure 6.18.Average and best optimal cost on Hanoi network under Univariate Gaussian Distribution with different number of components

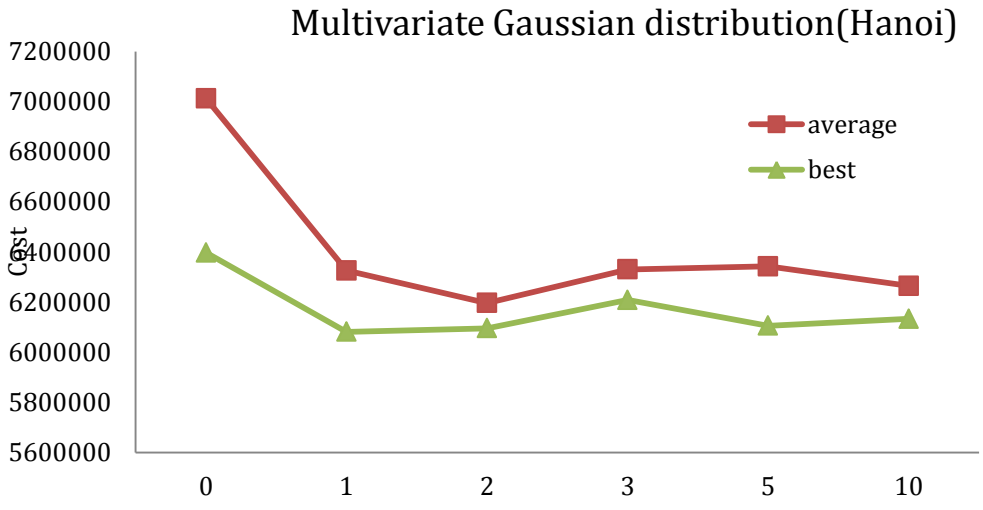


Figure 6.19.Average and best optimal cost on Hanoi network under Multivariate Gaussian

Distribution with different number of components

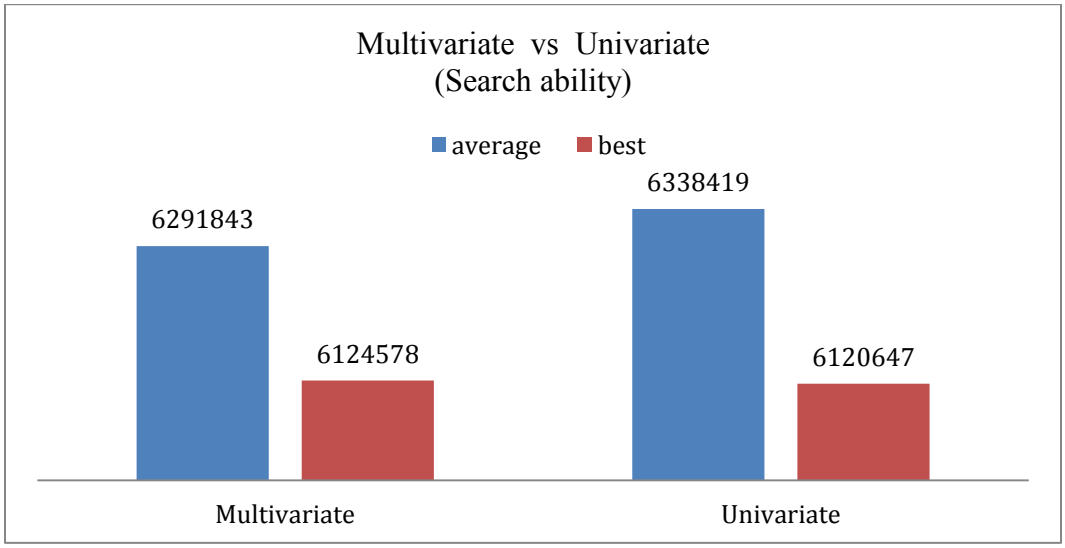


Figure 6.20.Compare the average performance of algorithms based on Univariate and Multivariate Gaussian models.

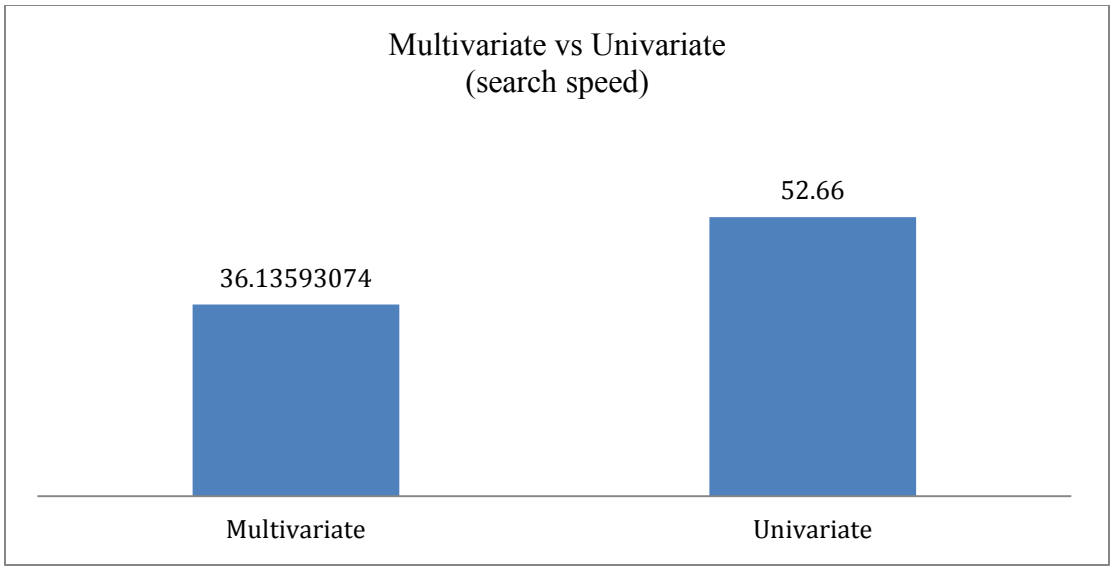


Figure 6.21.Compare the average number of generations before convergence based on Univariate and Multivariate Gaussian models.

In Figure 6.18 and Figure 6.19, the average and best optimal cost achieved by the proposed algorithms based on different assumptions are compared with that of conventional integer PSO. It is obvious that with the same population size and number of generations, both of the two proposed new methods performance much better than conventional PSO and this could prove that the proposed method for predicting promising region could effectively improve the search performance of PSO. On the other hand, on different number of Gaussian components, 2 or 3 components are better in terms of the average optimal cost, although there are similar results achieved by larger number of components. When compare the performance on different assumptions, from Figure 6.20, we can see that the average optimal cost of Multivariate model are better than Univariate Model. It might shows that fact that there are interactions between different pipes.

The Search speed (convergence rate) is also investigated and it is measured by the average number of generations before convergence within 100 generations. Figure 6.21 shows that the Algorithm based on Multivariate model converges much faster than that of Univariate model.

In sum, based on the experimental results on Hanoi network, we find that the assumption of Multivariate Gaussian model is more appropriate for Hanoi network than Univariate mode because the algorithms based on Multivariate model outperformances better in terms of search ability and search speed.

Due to the high dimensionality for the large network (Balerma), the proposed algorithms could not run on larger number of Gaussian components because the matrix manipulation crashed because of the singularity of the matrix data. Therefore, the algorithms only run on several number of Gaussian mixture models. The results are shown in f following table 6.11.

As listed in Table2, the performance of conventional PSO is terribly worse than the proposed algorithms, which could uncover the potential search ability of these proposed algorithms for larger size networks. The performance of UEPSO is generally better than that of MEPSO, especially the UEPSO under single Gaussian assumption. This implies that Univariate assumptions are more suitable for large size network, which might be due to the curse of high dimensionality.

Table 6.11.Experimental Results of EMPSO on Balerma network

Algorithms	# of	Average cost (Million	Minimal cost

	Gaussian Components	USD)	
IPSO	NA	102.4481	43.4391
UEPSO (Univariate)			
	2	9.175852	8.920233
	3	4.307014	3.584975
	>3	Not available (the matrix are singular)	
MEPSO(Multivariate)	1	3.74591	5.93152
	>1	Not available (the matrix are singular)	

6.2.8 Conclusions & Future Work

In this paper, two different strategies of predicting promising region using Expectation Maximization (EM) for PSO are proposed and compared. The implementation of prediction is based on two different assumptions about the distribution of good positions in the search space. The proposed algorithms are applied to the practical engineering optimization problem of urban water distribution network. Two famous benchmark examples are utilized to test the performance of the new algorithms and compared with previous work. Based on the experimental results so far, we can draw following conclusions:

1. Predict promising region using EM could greatly improve the search performance of PSO on the Water Distribution Network optimization problem.
2. Multivariate Gaussian Mixture Model is more suitable for small size network and Univariate Gaussian Mixture Model is more suitable for larges size network.
3. When used Gaussian Mixture Model, 2 or 3 Gaussian components are the best choice.

So far the experimental results has proven the proposed methods are much better than the conventional PSO and predicting promising region using Gaussian Mixture Model and EM could greatly improve search ability of PSO. In order to further verify the proposed methods, the following work needs to be done:

1. Compare the proposed methods with other methods other than PSO based algorithms on the same optimization problem.
2. Try to figure out a way to reduce the dimensionality of the network so that the methods cold be applied on larger size network.

6.3 Parameter Adaptation

6.3.1 Introduction of Parameter Adaptation

For most of the meta-heuristics, including PSO, performing the appropriate parameter adjustments is always a cumbersome and laborious task. Normally, meta-heuristics can only perform efficiently and effectively by adjusting the parameters properly. There are two primary ways of setting parameter values: parameter tuning and parameter control. Parameter tuning consists of finding and setting a priori parameter values before running, whereas parameter control is about adjusting the parameters as the algorithm is running. More specifically, three distinct ways of parameter control can be used: Deterministic parameter control, adaptive parameter control and self-adaptive parameter control (Figure 6.22). In deterministic control strategy, the parameters are altered by some deterministic rule, such as time varying strategy in which the parameter is varying over time following a predefined function of time. Adaptive parameter control takes place when there is some form of feedback from the search that is used to determine the change strategy of the parameters. Parameters are encoded into the

representation of the individuals in the population to co-evolve with the configurations of the individuals. The rationale behind the self-adaptive control is that the better values of the encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values. Adaptive parameter control is common for PSO and has shown effectiveness in solving various problems in the past [67]~[69]. It has also been used for WDN optimization problem. Montalvo [23] uses a self-adaptive parameter controlled PSO to optimize the WDS for the first time. The performance results of the proposed algorithm show that the self-adaptive featured PSO averages out the standard PSO and other EAs applied to the WDS optimization problem.

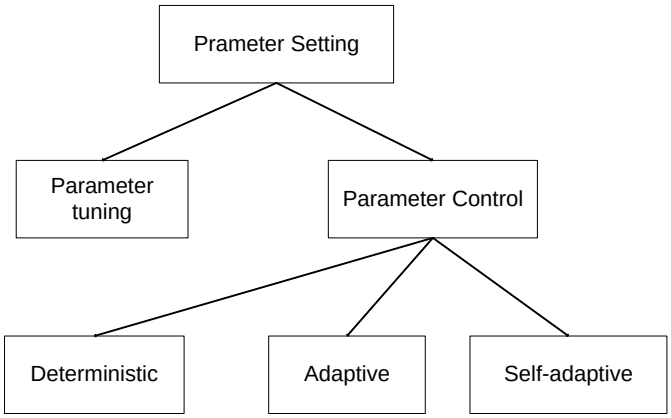


Figure 6.22.Classification of parameter control methods

6.3.2 Co-Evolution Strategy for Parameter Setting of PSO

In our proposed parameter estimation strategy, there are two iterative loops which run simultaneously. The first loop is PSO which tries to find the optimal solution for the specific problem; the second loop (coevolving loop) is the EDA which tries to optimize the parameter settings (c_1 and c_2) online for each individual in PSO loop. In PSO population, each individual is

a candidate solution for the specific optimization problem. In EDA population, each individual is a pair of values for c_1 and c_2. The population size for both PSO and EDA loop are the same and one individual in PSO corresponds to one individual in EDA population respectively. Each individual in PSO is updated based on (1) only using the corresponding pair of c_1 and c_2 in EDA poplation. Each individual in EDA population is updated every generation (see Algorithm 6.5). But this update is only implemented every M (1-10) generations in PSO loop, which means the parameter values for each individual in PSO keeps unchanged during every M consecutive PSO loop interactions. The fitness used in EDA loop is the progress of the particle under the corresponding parameter settings in PSO loop during M consecutive generations. Therefore, the values of c_1 and c_2 in EDA loop will hopefully converge to the best combination under which the corresponding particle in PSO could achieve the biggest progress in M consecutive generations. There is a assumption behind is that a good setting of parameters for the most recent iterations (e.g. past M iterations) will also be good (at least not worse) for the immediate future iterations. The complete pseudo code is given in following Algorithm 6.6.

Algorithm 6.5. Updating process in EDA Loop

1: t is the generation index of PSO loop;

2: x_i^t is the ith individual in P_{eda} at generation t;

3: y_i^t is the ith individual in P_{eda} at generation t;

4: N is populations size;

5: While i ≤ N do

6: f(y_i^t)= $f(x_i^t)$-f(x_i^{t-M});

7: i=i+1;

8: End While;

9: Rank P_{eda} according to fitness (decsending);

10: Select the top individuals from P_{eda};

11: Estimate the mean and standard divaton for each dimension;

12: Sample N individuals by the estimated distribution;

13: Update all the individuals in P_{eda} with the sample.

Algorithm 6.6. Parameter co-evolving algorithm

1: t is the generation index of PSO loop;

2: N is populations size of both P_{pso} and P_{eda};

3: Initialize the population for PSO: P_{pso};

4: Initialize the population for EDA: P_{eda};

5: Evaluate the each individual in population P_{pso};

6:While iter_number ≤ Max_iterations do

7: Update all the individuals in P_{pso} according to

8: (1)and (2);

9: If (iter_number)mod(M)=0

10: Update all the individuals in P_{eda}

11: according to Algorihtm 2;

12: End if ;

13:End While .

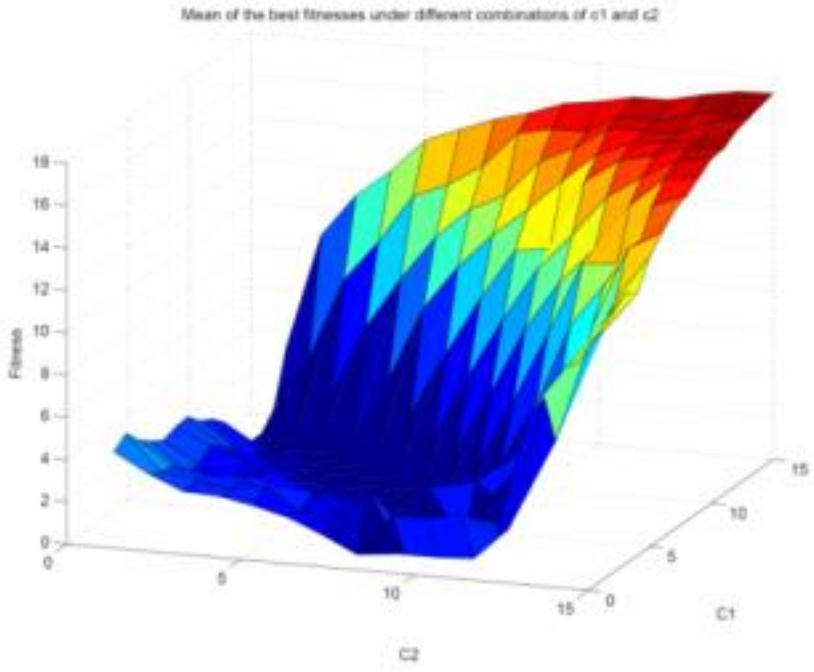
6.3.3 Validation by Numerical Optimization

The proposed parameter setting method is validated on two different numerical optimization

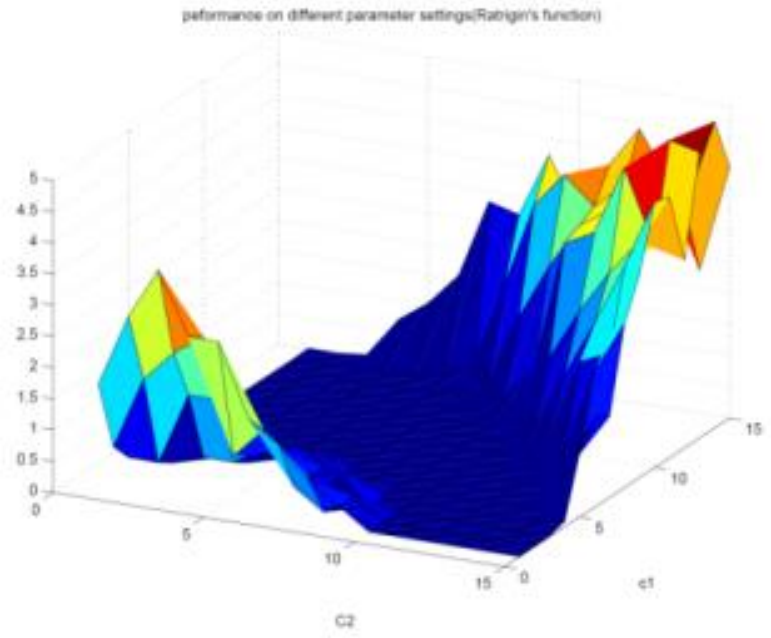
problems. The estimated optimal parameter settings (c1 and c_2) obtained by the proposed algorithm on these problems are compared with the values found by a fine parameter tuning. Ackly's function and Restringing's function are selected in this experiment due to their very rough landscapes with large numbers of local minima which makes any search algorithm vulnerable to being trapped in a local minimum. They are also usually used as benchmark problems for evaluating proposed new optimization algorithms.

1) Parameter Fine Tuning

A fine-tuning experiment was conducted to identify the optimal combination of the parameters c_1 and c_2. The best combinations for c_1 and c_2 were found to be [0.8, 1.8] for Ackley's function and [1.2, 1.6] for Ratrigin's function. Actually, from the 3D scatter plot in Figure 6.23, we can see that the best c_1 and c_2 combination is not just one point, but rather an area in the basin. In other words, there is a family of c_1 and c_2 combinations which could help the PSO algorithm perform the best.



Ackley's



Rastrigin's

Figure 6.23.Parameter tuning results for two numerical optimization problems

Table 6.12.The Tuning and Estimated Results for c1 and c2

parameters	**Tuned results**		**Estimated results and difference**	
	Ackley's function	**Rastrigin's function**	**Ackley's function**	**Rastrigin's function**
C_1	0.8	1.2	0.861752	1.121347 (-6.6%)

			(+7.72%)	
C_2	1.8	1.8	1.722622 (-4.3%)	1.665069 (-7.4%)

2) Parameter Estimation

For each of the problems, the above proposed parameter coevolving strategy is used to estimate the best parameter setting. And it was run 10 times. The average values for c_1 and c_2 are taken as the estimated result. From Figures 6.24 and 6.25 below, we can see that, for one particle, the c_1 and c_2 converged to almost fixed values at the end of the search process. Also from Figure 6.26, it is clear that for all the particles tracked, c_1 and c_2 converged to the same or nearly the same values respectively. Table 6.12 shows both fine-tuned and estimated best combination of c_1 and c_2 for each problem. The differences of between them are all less than 8%. Furthermore, in the 3D plot of the results of parameter tuning, there is an obvious basin which means three are more than one pair of good parameter settings and all the settings located in the basin can be regarded as optimal or near optimal settings. The estimated results are found to be located in the basin area. Therefore, we can say all the estimated results are generally consistent with the results of fine tuning.

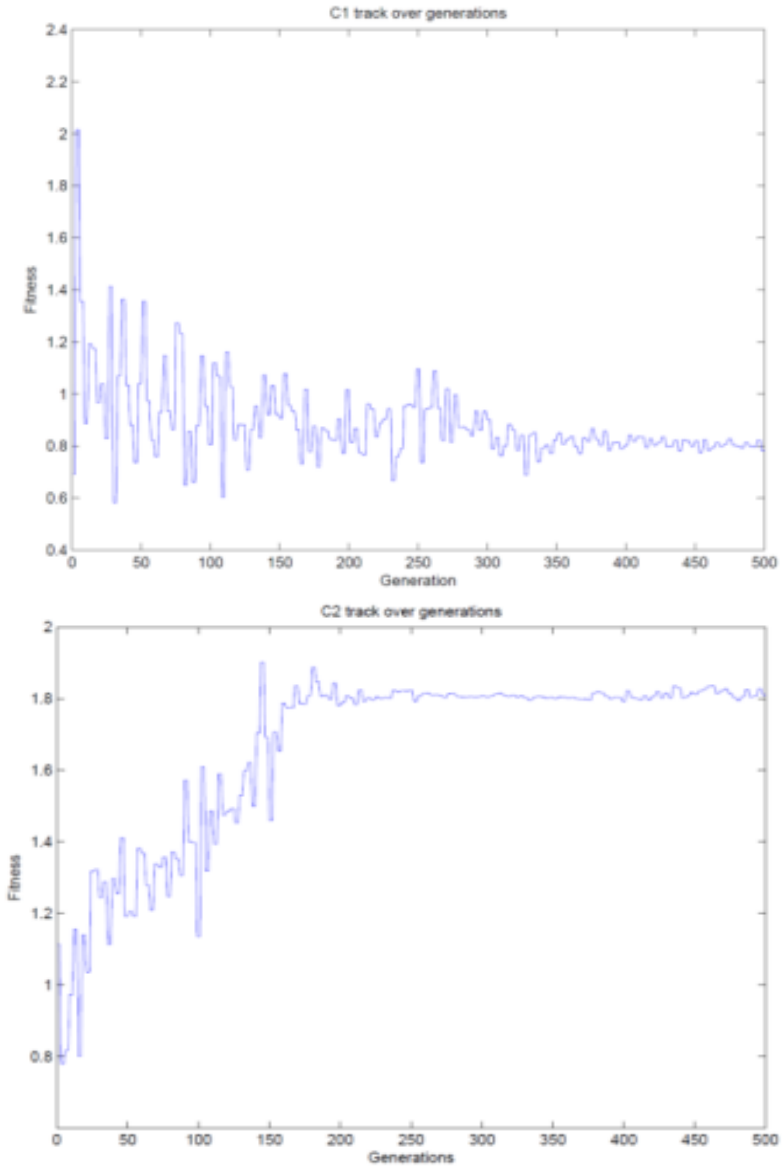


Figure 6.24.Track of c1 and c2 of one particle in one run (Ackley's function)

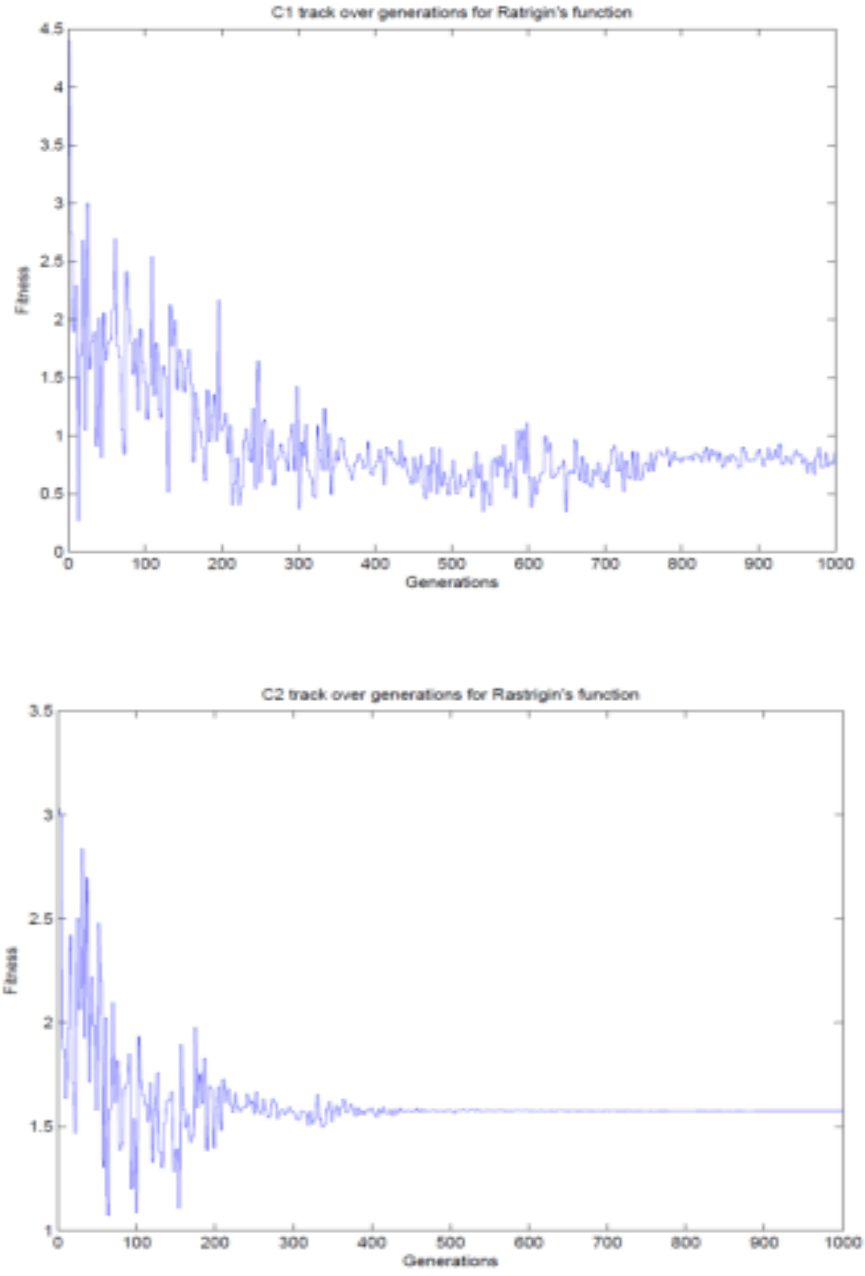


Figure 6.25.Track of c1 and c2 of one particle in one run (Rastrigin's function)

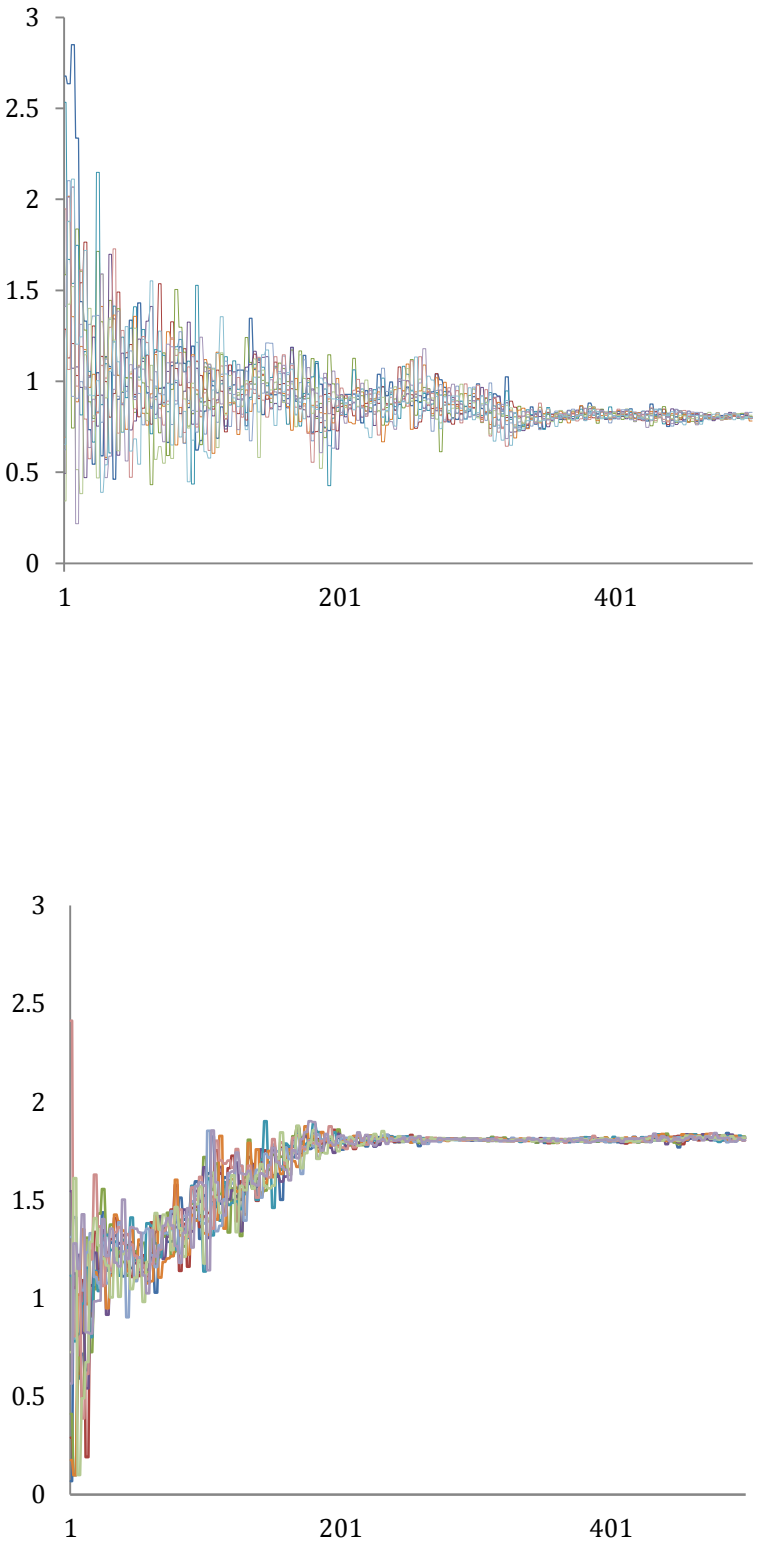


Figure 6.26.Track of c1 and c2 of 10 particles in one run (Ackley's function)

6.3.4 Benefits of the Fast Estimation Strategy

For most practical optimization problems, especially for engineering design optimization problems, exhaustive parameter tuning is very costly and probably intractable due to the resource limits (time and experimental cost). However, by using this fast parameter estimation strategy, an optimal parameter setting (or near optimal) can be identified in a very short time. Table 6.13 shows the average value of estimated values and Table 6.14 shows the time consumed by the two methods on different problems.

Table 6.13.Time Consumption by Two Different Methods on Different Problems

Parameters	Mean value	Standard Deviation
c_1	1.186576193	0.228964
c_2	2.057563122	0.423434

Table 6.14.Estimated Results for Hanoi Network

Problem	Tuning	Estimation	Time reduced
Ackley's function	2250 runs (37.5 hours)	10 runs (10 minutes)	99.5%
Rastrigin's	2250 runs (30 hours)	10 runs (8 minutes)	99.5%

6.3.5 Experimental Settings

In the implementation of PSO in this work, in addition to the two acceleration coefficients to be estimated by the proposed approach, there are two other parameters that need to be set. The first is the inertia weight over the generation which is computed using the following equation proposed by Jin et al.[70]:

$$\omega=0.5+\frac{0.5}{\ln(t)+1} \qquad (6.8)$$

Where: t is the iteration number. The boundary for the updated velocity (see Equation (1)) during the search process denoted by V_{max} also needs to be set. It is used to constrain the velocity of each particle to the range of $[-V_{max}, V_{max}]$. If V_{max} is too large, the search process will be close to a random walk which will make it very difficult to converge; on the other hand, if V_{max} is too small, the search process can easily be trapped in a local optimum. It is therefore important to set an appropriate value for V_{max}. Based on the findings by [23], the appropriate range for V_{max} is between 40% and 100% of the variable range. We use 50% in our experiments. Table 6.15 shows the details for the experimental settings:

Table 6.15.Experimental Setting.

Configuration	Hanoi Network
Dimensionality	34
Inertia weight	Varied according to equation (3)
Population size	200
Number of Iterations	500
Range of c_1 and c_2	[0,5] continues
NO. of runs	30
Implementation platform	Matlab2012

Table 6.16.Minimal Cost for the Hanoi Network

Methods	Minimal cost (x10^6\$)	Average (x10^6\$)	Average NO. of Fitness evaluations
GA[22]	6.093		
GA[23]	6.182		
GA[24]	6.195		
ACO[25]	6.367		
PSO[19]	6.133	6.487	80,000
PSO[20]	6.081	6.297	80,000
PSO[18]	6.081	>6.297*	80,000
PSO(this work)	6.081	6.252	20,000

*No specific number was given in the paper, but the authors compared their results in this paper with the results of their previous paper and found the average minimal costs are worse

6.3.6 Results & Discussion

The proposed estimation algorithm ran 30 times on the Hanoi network to estimate the optimal parameter settings for this engineering design problem. The results are presented in Table 6.16 and Figures 6.27 and 6.28 below.

As we can see from Figure 6.27, in each single run, there is a clear convergence of both c1 and c2 values during the co-evolution process. The average values of parameters in every run are recoded and the results of all 30 runs are displayed in Figure 6.28. We can see that all the average values are close, which means the estimated results are consistent throughout the experiments. The final average parameter values of the 30 runs are shown in Table 6.13, which is used as the estimated optimal parameter setting in the subsequent experiments.

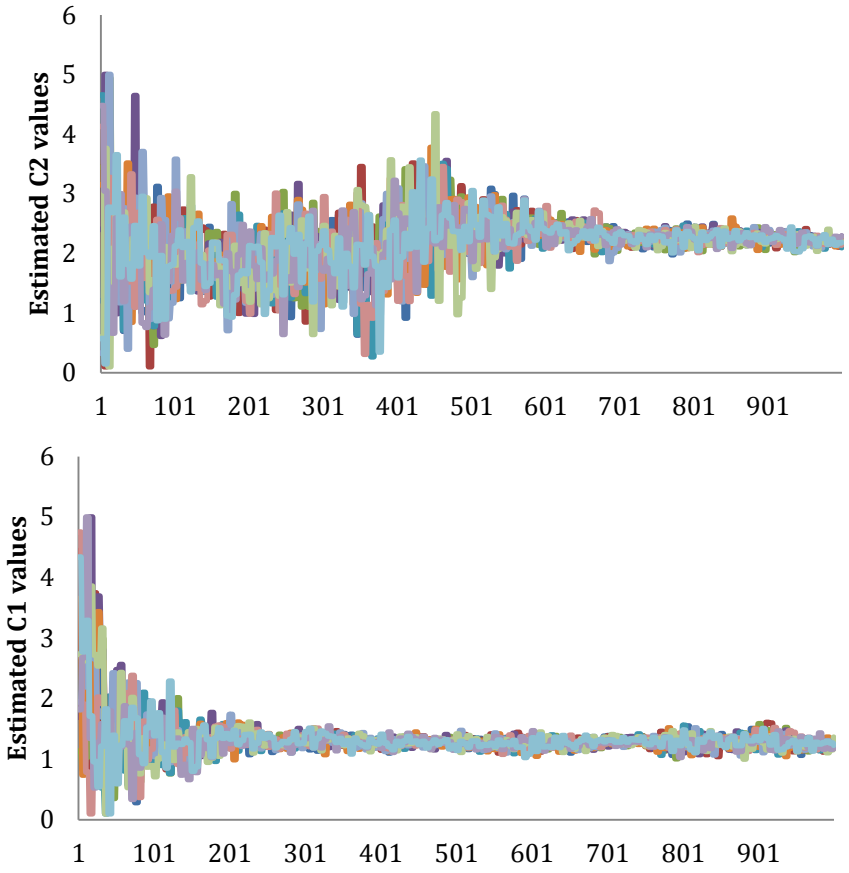
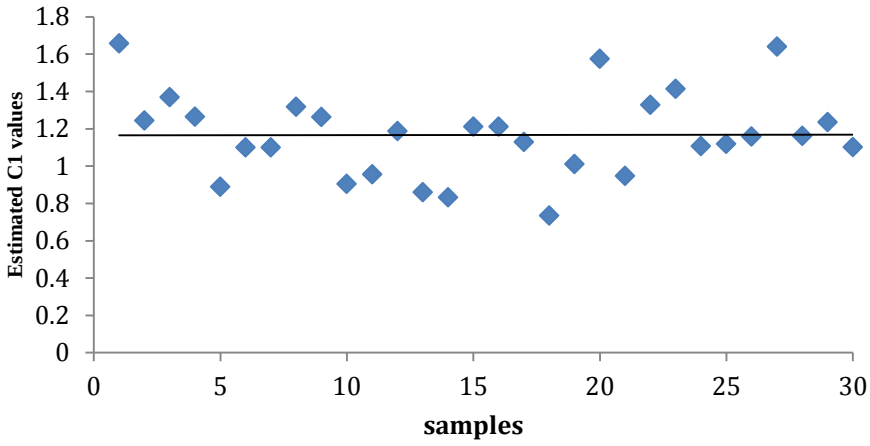


Figure 6.27.Track of estimated C1 and C2 values for 30 particles on Hanoi network



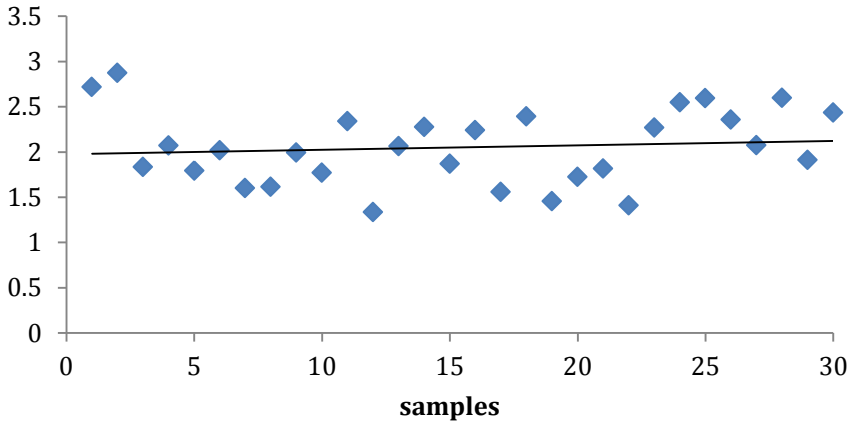


Figure 6.28.Estimated C1 and C2 values for Hanoi network

With the estimated optimal parameter settings: c1= 1.1866, c2=2.0576, we run the PSO for 30 times on the Hanoi network and the results are summarized in Table 6.16 above. An extensive comparison between the proposed algorithm and other methods from the literature is made. Our PSO with the estimated optimal parameter settings achieved the best known minimal cost which is 6.0811(unit price).

Although the best optimal solution known in the literature is only achieved once in the 30 runs (See Table 6.16), the reliability and efficiency are comparable to other methods (Table 6.16). The reliability of an optimization method could be measured by the average performance achieved by the proposed algorithm in a certain number of runs. Herein, we use the average minimal cost as the indicator of reliability. We know from Table 6 that our proposed algorithm achieved the smallest average cost compared to the other methods which also achieved the best minimal cost. Efficiency is measured by the average number of fitness calls used by the algorithm to achieve the best solution because in most engineering applications, fitness evaluation is the most time

consuming part of the optimization algorithm. Obviously, our method achieves the best efficiency, which is almost 4 times better than other methods. For the sake of clarity, in order to make the comparison justifiable, we need to point out that, except for the two adjusted parameters, all other parameters (e.g. inertia weight and V_{max}) are set to the same values used in the three previous works in which PSO was adopted. The 30 minimal costs are shown in a histogram in Figure 6.29 and we can see that most of the achieved minimal costs are close to 6.252 (unit price). The fitness traces of 5 runs are shown in Figure 6.30, from which a very fast convergence process can be observed. Another important observation from this figure is that all the 5 search processes converged after around 100 iterations. This might indicate that we could achieve a very good result with a very small number of fitness evaluations (around 2,000). This is a clear advantage over all other methods. From a practical point of view, an "early" almost-optimal solution may be preferred to a "very late" optimal solution when the cost of time is taken into consideration. On the other hand, this observation also suggests that there might be a premature convergence problem, which is a common problem with the application of PSOs. Several researchers have proposed methods to address this problem with no conclusive solution. The major task for our proposed estimation method is to help the PSO obtain a very good parameter setting in a very short time, but it is not able to tackle the premature convergence problem, so more work is required to further improve the performance of PSO.

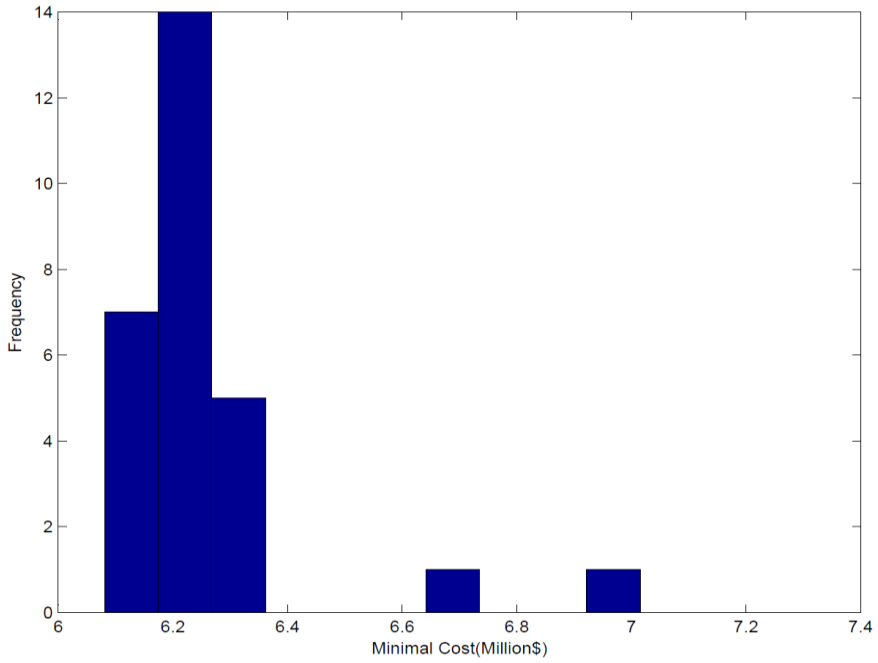


Figure 6.29.Minimal costs achieved with the estimated

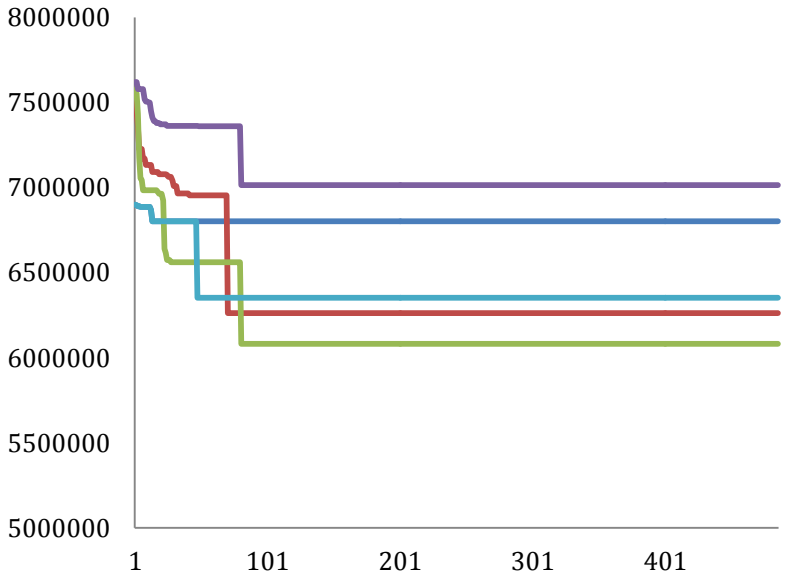


Figure 6.30.Fitness track of 5 runs

6.3.7 Conclusions and Future Work

In this work, a fast parameter estimation strategy for PSO is proposed to estimate the optimal acceleration parameter settings. The self-adaptive control and co-evolution strategies are used together for the parameter estimation of PSO. In the proposed estimation model, the PSO is used to search the optimal solution for the specific problem while an Estimation Distribution Algorithm is used to adjust two acceleration coefficients iteratively according to the performance of their settings. In such way, acceleration coefficients of each particle are co-evolved throughout the search process. The tests on two numerical optimization problems show that the estimated optimal parameter settings are consistent with the optimal parameter settings achieved by parameter tuning. The application to a WDN optimal design problem shows that the proposed algorithm successfully estimated good parameter settings with which the PSO was able to identify the best known optimal solution for the benchmark example. The algorithm also performs better than other methods in the literature in terms of reliability and efficiency, although there is a premature convergence problem. The proposed parameter estimation strategy is promising and may perform better when combined with a premature convergence elimination strategy. The major contributions of the work presented in this paper are as follows: (1) A novel fast parameter estimation strategy to replace the inefficient conventional parameter tuning method for PSO is proposed. As long as a good parameter setting could be identified for a specific problem, it is not difficult for the optimization algorithm to achieve a good solution. And(2), a new optimization method with PSO is proposed for the optimal design of a Water Distribution Network. It is reliably able to achieve a very good solution with a small number of fitness evaluations.

Our future work will focus on the following topics: (1) to extend the proposed fast parameter estimation strategy for PSO to other Evolutionary Algorithms where there is a need for parameter tuning. And (2) to apply the proposed algorithm to the design optimization of larger water distribution networks design or other practical engineering design and/or optimization problems.

6.4 Operator Self-adaptation for PSO

6.4.1 Operator Adaptive Strategies

Adaptation is one promising research trend in PSO. Many adaptive and self-adaptive strategies are used to improve the performance of PSO. However, most of them focus on the parameter adaptation, only a few have focuses on the operator adaptation mechanism. In conventional PSO, all the particles are updated by the same strategy, in other way, all particles behave according to the same strategy. While as a matter of fact, there is no single operator is optimal for all the problems all the time. Therefore, each particle has several operators for selection. They can select the most suitable operator to update its velocity according to the environment or landscape. And the most important thing is that all the particles in the swarm have the ability of sensing the environment in the landscape of the search space and act independently upon what situation they encounter and where they are in the landscape. That is to say, each particle use different updating strategy at different time intervals and all these particles are using different updating strategies at the same time instant. The hypothesis of this method is that each particle can be regarded as an agent and can own a certain level of intelligence because it possesses the basic characters of an agent: perceiving its environment and acting upon that environment through actuators. From an agent's point of view, the environment for each particle agent in the

search landscape can be regarded as partially observable, stochastic, episodic and cooperative multi-agent environment. The philosophy behind this proposed model is that when each particle in the swarm has some kind of intelligence, the overall performance and search quality of the whole swarm will be improved.

This type of adaptation mechanism based hyper heuristic was investigated in [71]. Based on the fact that no single operator is optimal for all problems and the optimal choice of operators for a given problem is also time-variant, Smith and Fogarty [72] proposes a framework for the classification based on the learning strategy used to control them and reviewed a number of adaptation methods in GAs. They also addresses the issue that the set of available operators may change over time, Smith [73] proposed a method for estimating an operator's current utility, which is able to avoid some of the problems of noise inherent in simpler schemes for memetic algorithms. In [74], [75], an adaptive allocation strategy, called the adaptive pursuit method, was proposed and compared with some other probability matching approaches in a controlled, dynamic environment. In [76], a specific dynamic method based on the multi-armed bandit paradigm was developed for dynamic frameworks. In order to well evaluate the performance of operators, recently, a new strategy [77] was introduced by considering not only the fitness improvements from parent to offspring, but also the way they modify the diversity of the population, and their execution time.

Based on abovementioned analysis, in this proposed operator self-adaptation strategy (OSPSO), there are two issues we need to solve: a set of effective operators which could the particles learn from different information source to improve the fitness of the particles, and a credit assignment system which forms the operator selection mechanism that automatically select the best operator

for each particle at different stage of the search process. In the following two sections, we will discuss about how to solve these two problems.

6.4.2 Learning Operators

In this proposed operator self-adaptation PSO (OSPSO), three operators are designed for each particle so that they could independently deal with different situations and select the most suitable operator accordingly. The equations for each of the proposed operator are listed below:

(3) Operator a: learning from its own best position (Exploitation)

$$v_k^d = wv_k^d + \alpha * r_k^d(pbest_k^d - x_k^d) \tag{6.9}$$

(4) Operator b: Learning from the global best position (Exploration)

$$v_k^d = wv_k^d + \alpha * r_k^d(gbest_k^d - x_k^d) \tag{6.10}$$

(5) Operator c: Learning from good positions in the history (Prediction)

$$p_{k+1}^d = mean(p_k^d) + std(p_k^d) * randn() \tag{6.11}$$

Where *k* is the iteration number; *d* is the dimension index; $pbest$ is the personal best position of each particle; *gbest* is the global best position of the entire swarm; *randn()*is a random standard normal distribution. $mean(p_k^d)$ and $std(p_k^d)$ are the mean value and standard deviation of each dimension of better half of personal best positions in the current population.

As we can see, the first two operators are directly introduced from the updating strategy of original PSO. But the last one is a new operator which could learn the useful information from

the historical good positions the particles have ever traveled. This information is recorded in the personal best positions of the current population during the search process.

6.4.3 Operator Selection Mechanism

After the operators are designed, the next issue is about how to select the best one from them which involves the operator selection mechanism. There is a very important assumption for the operator selection mechanism which is that good operators in the previous several iterations are also good choice for the particle in the immediate successive following iterations. The key function of operator selection mechanism is to assign the credit for each operator as reward according to its performance in previous implementations. Our basic idea is to assign each operator for each particle a selection probability. And the selection probability will be updated every certain iterations of the search process according to the performance of each operator in previous iterations. There are two aspects of performance of the operator should be taken into consideration:

(1) The successfully ratio: which means how many time the operator successfully improve the fitness of the particle when the operator is selected.

(2) The progress achieved: which means how big the improvement achieved by applying the operator on the particle.

Beside above two performance metrics, the selection probability in previous iterations should also be included into consideration of assigning the reward of each operator. The reward assignment mechanism is defined as

$$r_i^k = \frac{p_i^k(k)}{\sum_{j=1}^{3} p_i^k(k)} * \beta + \frac{g_i^k}{G_i^k}(1-\beta) + c_i^k s_i^k(t) \quad (6.12)$$

$$c_i^k = \begin{cases} 0.8, & if\ g_i^k = 0\ and\ s_i^k = \max(p_i^k) \\ 1, & otherwise \end{cases} \quad (6.13)$$

Where g_i^k is the counter that records the number of successful learning times of particle k, in which its child is fitter than particle k by applying operator i since the last selection ratio update, G_i^k is the total number of iterations where operator i is selected by particle k since the last selection ratio update, $\frac{g_i^k}{G_i^k}$ is the success ratio of operator i for particle k, β is a random weight between 0.0 and 1.0, in this research, we set it as 0.5 because we give the same importance of successful ratio and the progress made by the operators. c_i^k is a penalty factor for operator i of particle k. Please note in (6.13), we are trying to give penalty for the operator which fails to make any improvement of the particle but it has the biggest selection ratio in the previous iterations. This is to avoid the situation that a good operator the past will still be in the domination position when the environment has completely changed. The full details of the OSPSO are given in following Algorithm 6.7.

Algorithm 6.7. Operator Self-adaptation PSO (OSPSO)

1: Initialize population; Selection probability update frequency=M.

2: Initialize selection probability of each operator for each particle in the population.

3: While (i< Max_iteration) do

4: Select one operator for each particle according to the selections probabilities.

5: Update each particle using the selected operator.(equaiton6.9,10,11)

If mod(i, M)==0

Update the slection probability of each operator for each particle according to equaitons 6.12 and 6.13.

End if.

6: Go to line 4.

7: End while.

6.4.4 Experimental Setting and Parameter Tuning

To test the performance of the proposed OSPSO on the optimization of WDN, it is used to optimize the two benchmark examples. At the very beginning, there is a very important parameter need to be tuned: update frequency of selection probability. So before the algorithm is tested on both the benchmark examples, it is fine-tuned and the results are shown in Figure 6.31 and 6.32. As we can see, the best update frequency for different network is not the same, for Hanoi network, the best frequency is 5 but for the larger Balerma network, it is 30. Based on these results, the parameter setting for the experiments are listed in the following table 2

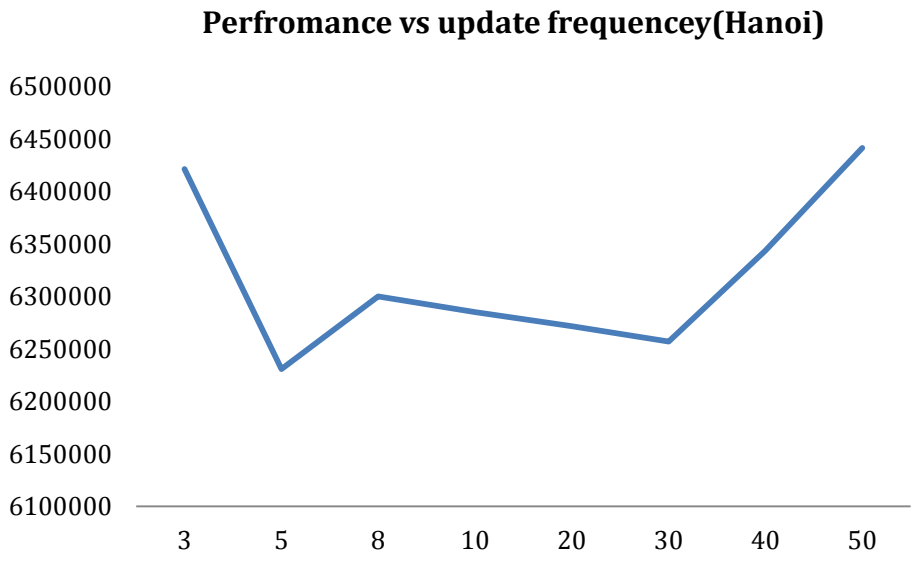


Figure 6.31.Performance of OSPSO on different update frequency on Hanoi network

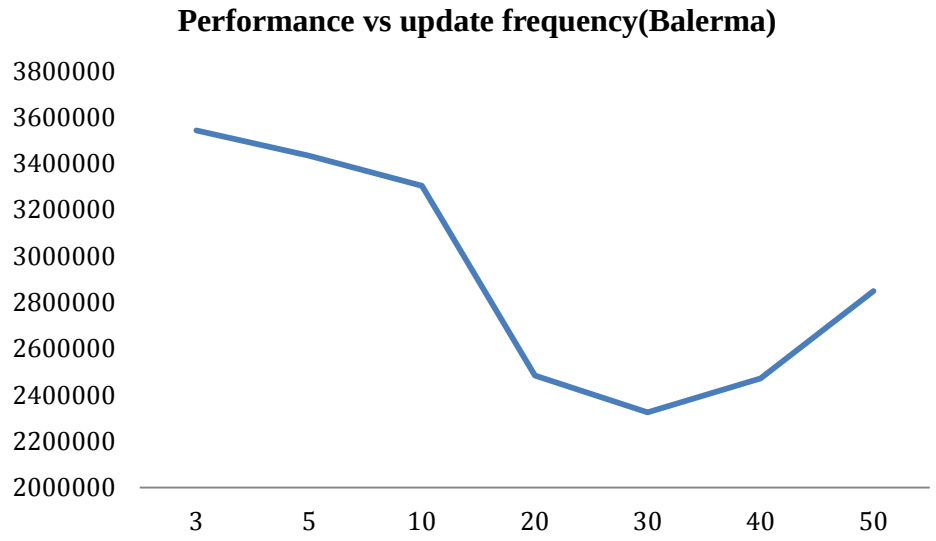


Figure 6.32.Performance of OSPSO on different update

Table 6.17.Parameter Setting for OSPSO

Network	Population size	iterations	NO. of runs	Update Frequency	α	β	w
Hanoi	500	500	30	5	2	0.5	1
Balerma	1000	1000	30	30	2	0.5	1

6.4.5 Experimental Results and Analysis

Using the parameter setting in table 6.17, OSPSO is applied to both of the benchmark networks and the results on Hanoi network are shown in Figure6.33 to Figure 6.35.

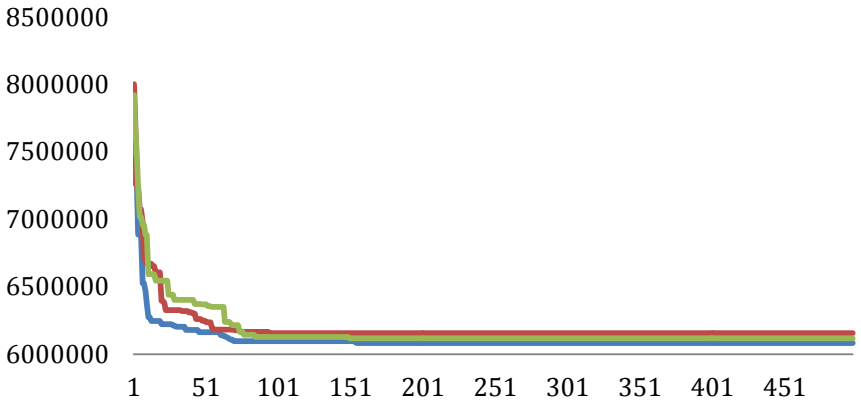


Figure 6.33.Fitness track of 3 runs of OSPSO on Hanoi network

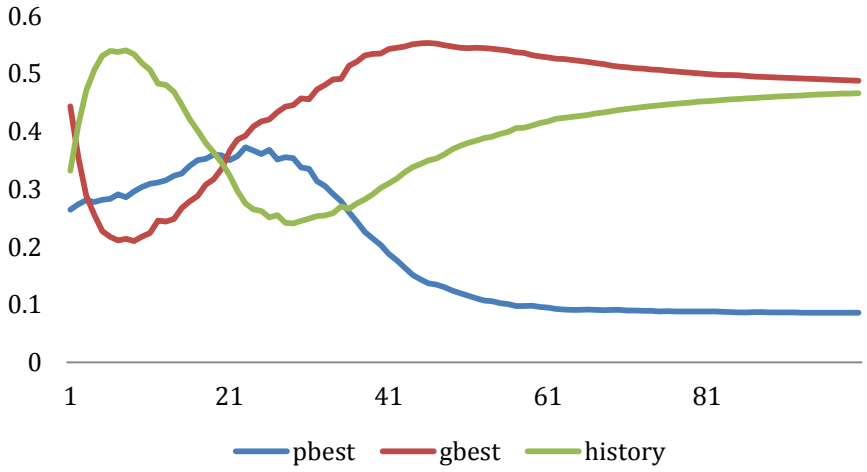


Figure 6.34.Track of average selection probability for each operator of OSPSO on Hanoi network

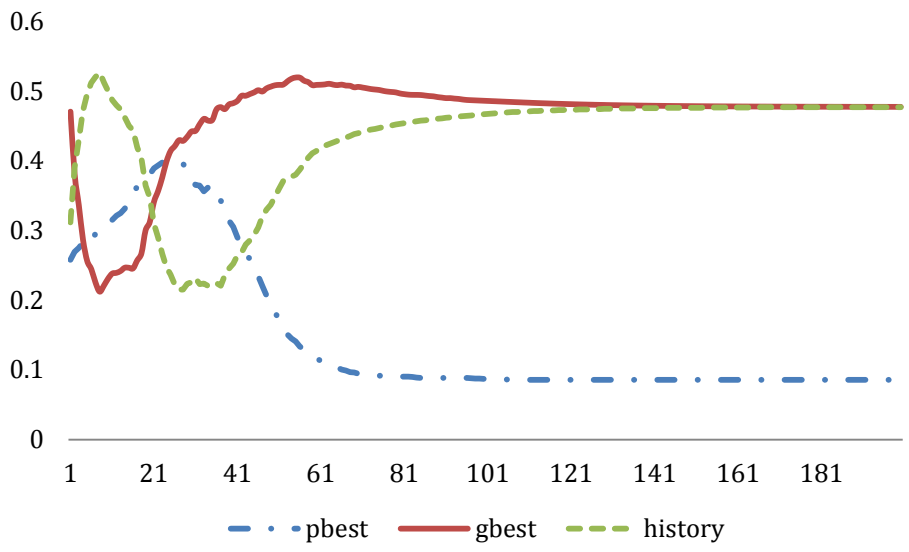


Figure 6.35.Track of average selection probability for each operator of OSPSO on Hanoi network(#2)

Figure 6.34 and 6.35 provide two examples of tracking the average probabilities of each operator throughout the search process. There is a very obvious fixed pattern of change of the probability for each operator. For the operator which learns from personal best position, its selection probability increase at the early stage but decreases as the population is close to convergence, which can be explained by the fact that at the ealy stage of the search process, many particles could move toward a local optima very quickly by learning from its personal best position which is usually an local optima, however, when the populations is close to convergence, the particle will be trapped in a local optima if it only learns from the local optima. On the contrary, the selection probability for operator learning from the global best positions is very high at the early stage, but it starts to decrease after about 10^{th} update. This decrease is not because learning from global best position does not work anymore, but because the selection probability of the third operator which learns from the historical good positions increase very fast (please note, the sum

of selection probability of these three operators is equal to 1). At the the very early stage, all the personal best positions are more scattered in the search space, so the distribution model built on these positions is not accurate enough, while as all the personal best positions are close to global optima, learning from these good positions will likely to find a better position in the landscape. Therefore, as the population approaches convergence, the selection probability of the third operator first decrees and then keeps increasing until it completely lose its function when the population is completely got convergent. And at last, all the operators will lose their function together and the selection probability will not change any more. In Figure 6.33, we can see that, at about 200th iteration, the search process got convergent, as the selection probability updating frequency is 5, so in figure 6.34, 6.35 we can see the selection probability of all the three operators got convergent after about 40^{th} updating times.

Now, let's look at the same experiment on the Balerma network that has different search space landscape.

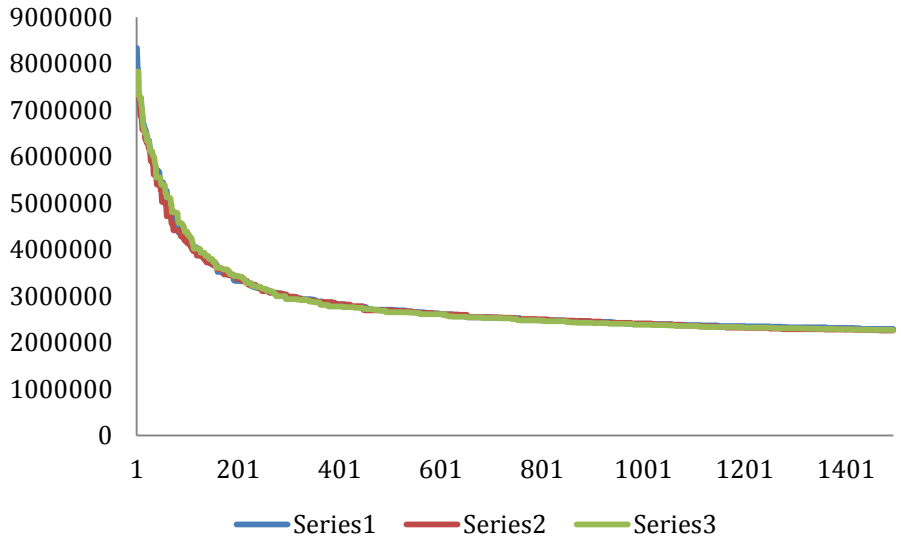


Figure 6.36.Fitness track of 3 runs of OSPSO on Balerma network

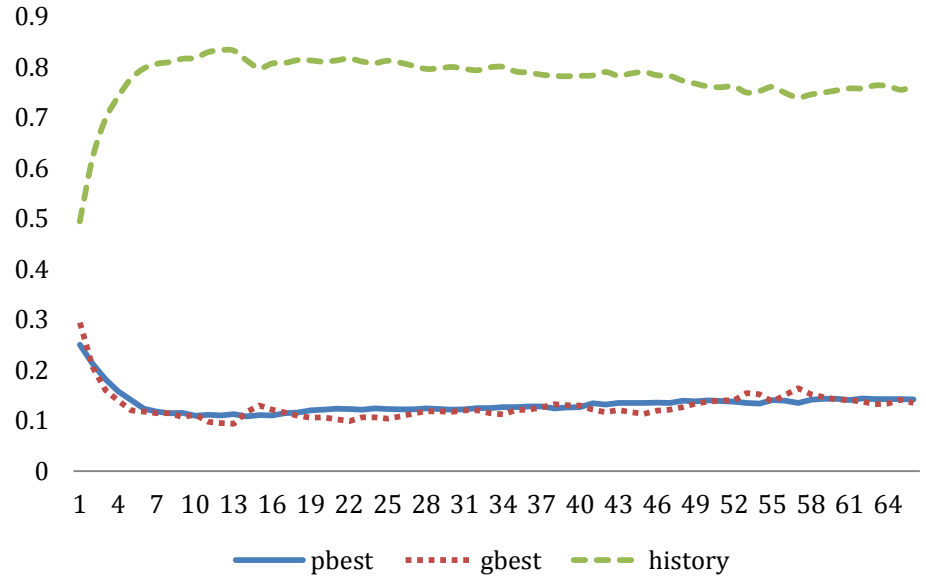


Figure 6.37.Track of average selection probability for each operator of OSPSO on Balerma network(#1)

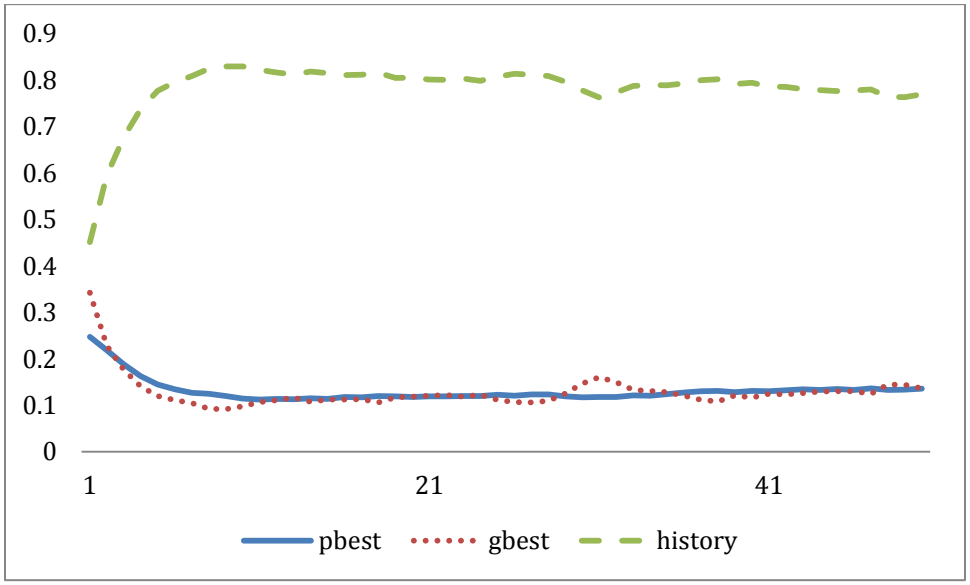


Figure 6.38.Track of average selection probability for each operator of OSPSO on Balerma network(#2)

As we can see in figure 6.37 and figure 6.38, the change trend of all the operators is completely different from that on Hanoi network. This is mainly due to the reason that the search space landscape is completely different form that of Hanoi network, on the other hand, this could also verify our assumption that each particle could select the right operator according to its position in the landscape. As we can see, at the early stage, the operators learning from personal best and global best positions lose their function very quickly and the selection operator learning from historical best positions dominates the two others very quickly. What is notable is that this result gives a perfect explanation why the conventional integer PSO performances very badly on Balerma network (see Figure 7.2), because in conventional integer PSO, only the operators learning from personal best and global best positions used. Additionally, it could also explain why SEDPSO and PEDPSO could both perform well on Balerma network (also see figure 7.2), because the process of learning from historical good positions are also integrated into the search process. The best achieved minimal costs are recorded in table 6.18.

Table 6.18.Experimental results of OSPSO.

network	Population size	iterations	NO. of runs	Update Frequency	Best cost	Average cost	Standard deviation
Hanoi	500	500	30	5	$6.081x10^6$	$6.231x10^6$	$1.332x10^5$
Balerma	1000	1000	30	30	$2.256x10^6$	$2.325x10^6$	$8.958x10^4$

6.4.6 Conclusions and Future work

This proposed operator self-adaptive PSO is very efficient PSO variant for the problem. The experimental results could explain some theoretical analysis of the PSO performance.

Tis should be very promising research spot and following directions could be immediately figured out:

1). Create more operators to learn from different information source. For instance, we can introduce some operators from other swarm intelligence algorithms to this operator self-adaptive algorithm framework (e.g. instinctive movement operator from Fish school search).

2). More efficient decision making strategy could be introduced to implement the procedure of selecting the most suitable operator.

6.5 Hybridized with a Local Search algorithm

6.5.1 Hybrid Evolutionary Algorithms

As we know, some evolutionary algorithms (e.g. Genetic Algorithm) are good at exploring the search space while others (e.g. Iterated Local Search) performs better in exploiting the local search space. So many researchers try to hybridize different type of EAs. The hybrid algorithms attempt to obtain the best from the hybridization of classical evolutionary search algorithms that perform together and complement each other to produce a new efficient algorithm modal.

In this chapter, a new hybrid algorithm that consists of PSO and EO is proposed for WDN optimization problem.

6.5.2 Extreme Optimization

Extreme optimization (EO) is a general-purpose local search heuristic based on the understanding of the far-from-equilibrium phenomena in terms of self-organized criticality (SOC) [P.Bak,1987]. EO appears to be a powerful addition to the traditional Meta-heuristics (e.g.

Genetic Algorithm) in its generality and its ability to explore complicated configuration spaces efficiently. The major feature of EO is that it is not population based and during the iterations the worst components of the representation is continually identified and replaced. Following Algorithm 6.8 is a basic model of EO.

Algorithm 6.8 Basic EO model

1: Initialize the configuration S at will; set S_{best} = S.

2: For the "current" configuration S,

(a) Evaluate each variable in S and give them a fitness value: for each variable in S, we keep other variable constant and then increase and decrease by 1 to see in which direction the entire S will be improved largely. At last, record the direction and fitness value for each variable in S.

(b) Sort the variables according to the fitness value and find the largest fitness and corresponding direction. This variable can be regarded as the worst variable x_j

(c) Change the variable x_j randomly in its corresponding direction and obtain a new solution S' in the neighborhood of S.

(d) Accept S' = S unconditionally.

(e) If C(S)< C(S_{best}) t hen set S_{best}=S

3: Repeat at step 2 as long as desired

4:Return S_{best} and C (S_{best})

To avoid the situation in which the original EO is likely to be trapped in a deterministic process: selecting always the worst variable in step (2b) which constrains the diversity of the selection results. So, a stochastic process is added into the algorithm. The improved algorithm is implemented as following:

Algorithm 6.9. Improved EO model

1 Initialize the configuration S at will; set S_{best} = S.

2 For the "current" configuration S,

(a) Evaluate each variable in S and give them a fitness value: for each variable in S, we keep other variable constant and then increase and decrease by 1 to see in which direction the entire S will be improved largely. At last, record the direction and fitness value for each variable in S.

(b) Sort the variables according to the fitness value and find the largest fitness and corresponding direction. Consider a probability distribution over the ranks k, and then select x_j according toP_k.

$$P_k \propto K^{-t}$$

(c) Change the variable x_j randomly in its corresponding direction and obtain a new solution S' in the neighborhood of S.

(d) Accept S' = S unconditionally.

(e) If C(S)< C(S_{best}) t hen set S_{best}=S

3 Repeat at step 2 as long as desired

4 Return S_{best} and C (S_{best})

In the proposed hybrid algorithm, EO will be used as a local search strategy integrated into PSO to enhance the exploitation ability of the search.

6.5.3 Test on Hanoi network

For Hanoi network, we use EO algorithm to update all the particles every certain number of PSO

iterations. The local search frequency (LSF) is a very essential parameter needs to be fine-tuned. We set the LSF as 5, 10, 20 and 40. As can be seen in Figure 6.39, 5 is the best local search frequency, which means use EO to improve all the particles every 5 PSO iterations. Figure 6.40 gives the fitness track, we can see that there is no clear premature convergence problem compared to convention integer PSO without local search process.

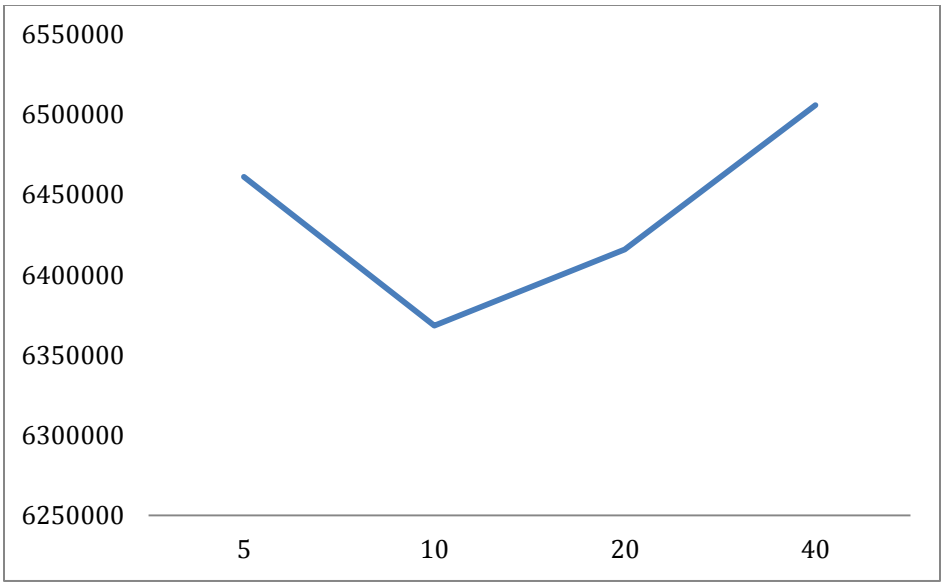


Figure 6.39. Performance at different local search frequency

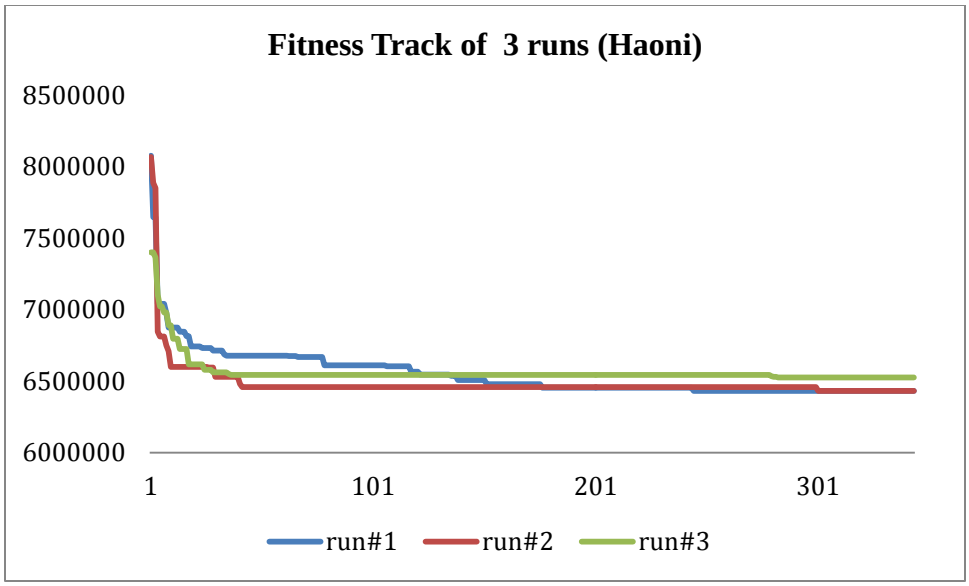


Figure 6.40.Fitness track of 3 runs on Hanoi network

6.5.4 Test on Balerma Network

On Balerma network, different strategy is used. Since the number of fitness evaluations used for one single EO process is the number of pipes and the populations size is also proportional to the network size, EO is not used to update every particle in the population, instead, it is used only to improve the global best position every PSO. Figure 6.41 and Table 6.19 are showing the final results on Balerma network.

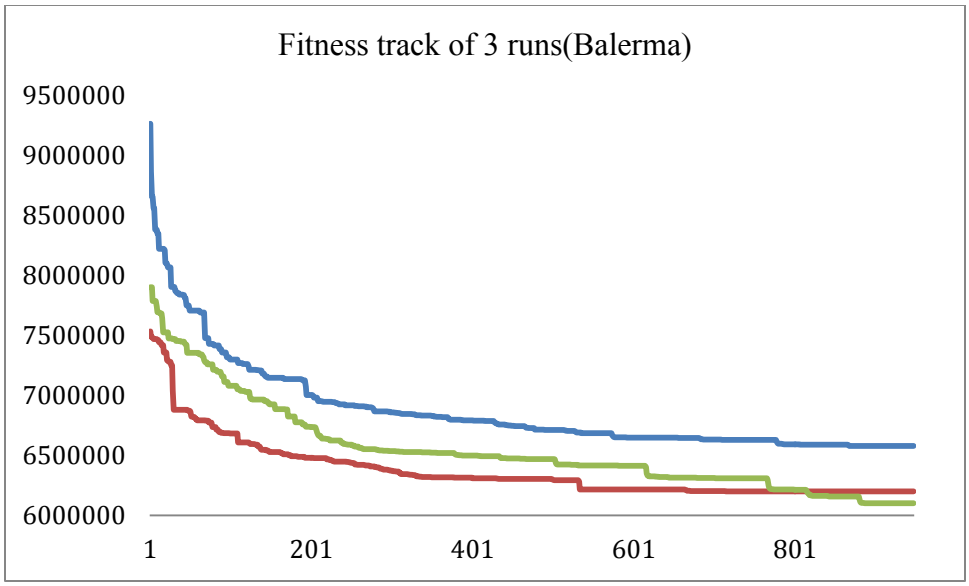


Figure 6.41.Fitness track of 3 runs on Balerma network

Table 6.19.Experimental Results of PSO-EO

Network	Population size	Maxim #NO. of generations	c1,c2,w	LSF	Best cost	Average cost	Standard deviation
Hanoi	500	500	2,2,0.8	10	$6.198x10^6$	$6.368x10^6$	$1.495x10^5$
Balerma	1000	1000	2,2,0.8	1	$6.046x10^6$	$6.734x10^6$	$6.543x10^5$

CHAPTER 7

7 PERFORMANCE COMPARISON

7.1 Summary of Proposed Algorithms

Eleven different Algorithms are proposed for optimization of Urban Water Distribution network in this thesis research. For the sake of easy comparison, we give each algorithm an abbreviation and index which will be used in the following tables and figures. Their descriptions are listed in following Table 7.1:

Table 7.1.Index and Abbreviation of Proposed Algorithms

Index	Abbreviation	Description	Chapter
1	SPSO	Standard Integer PSO	5.1
2	PE- PSO	Parameter Fast Estimation PSO	6.3
3	ISEDPSO	Sequential EDA enhanced PSO	6.1.3
4	PEDPSO	Parallel EDA enhanced PSO	6.1.3
5	PSO-EO	EO enhanced PSO	6.4
6	OS-PSO	Operator self adaptation PSO	6.4
7	Shuffle-PSO	PSO with shuffle process	6.1.2
8	EMPSO-U	Expectation Maximization enhanced PSO using Univariate Gaussian mixtures	6.2

9	EMPSO-M	Expectation Maximization enhanced PSO using Multivariate Gaussian mixtures	6.2
10	FSS	Original Fish School search	5.2
11	CFSS	Cluster Analysis enhanced FSS	6.1.4

7.2 Performance Evaluation Metrics

The performances of evolutionary algorithms are usually measured by the following famous metrics in the literature:

1. ***Success rate***: when the optimal solution is already known for a specific problem, the percentage of runs that successfully achieve the already known global optima. This metric is mainly used to measure the search ability of the algorithm of interest.
2. ***Number of evaluations***: the number of evaluations for an algorithm to achieve a target solution of predefined quality. This metric is majorly used to measure the efficiency of the algorithm of interest.
3. ***Reliability****:* it is measured by the standard deviation of the best solutions that can be achieved by the algorithm in a number of runs.

7.3 Performance Comparison

According to above performance metrics used in the literature, the corresponding values are

receded in Table 7.2: the best cost (minimal cost) that could be achieved in 30 runs; the average best cost of 30 runs; and the standard deviation of the best achieved cost in 30 runs. Since the average best cost is the most suitable indicator of the overall performance of the algorithm, all the algorithms are compared by the average best cost in Figure 7.1. As we can see, for Hanoi network, SPSO, FSS and CFSS are the much worse than the others. EMSPSO-M, PEDPSO and ISPEDPSO are the top three algorithms. However, for Balerma network, the situation is completely different. PSO-EO is the worst one and there are three algorithms much better than the others, they are PEDPSO, OS-PSO and ISEDPSO. The results show that different algorithms perform differently on the two benchmark networks. And from table 7.4, we know that only three algorithms performance well on both networks, they are PEDPSO, ISEDPSO and OS-PSO.

Table 7.2.Performance of the Proposed Algorithms on Hanoi Network

Index	Algorithm	Population size	NO. of iterations	Number of fitness evaluation	Best cost	Average best cost	Standard deviation
1	SPSO	500	500	250,000	$6.369x10^6$	$7.110x10^6$	$1.557x10^6$
2	PE-PSO	500	500	250,000	$6.081x10^6$	$6.252x10^6$	$1.317x10^5$
3	ISEDPSO	500	500	375,000	$6.081x10^6$	$6.102x10^6$	$9.441x10^4$
4	PEDPSO	500	500	250,000	$6.081x10^6$	$6.103x10^6$	$7.288x10^4$
5	PSO-EO	500	500	1950,000	$6.198x10^6$	$6.368x10^6$	$1.495x10^5$
6	OS-PSO	500	500	250,000	$6.081x10^6$	$6.231x10^6$	$1.332x10^5$
7	Shuffle-PSO	500	500	250,000	$6.081x10^6$	$6.299x10^6$	$1.140x10^5$
8	EMPSO-U	500	500	250,000	$6.236x10^6$	$6.262x10^6$	$1.151x10^6$
9	EMPSO-M	500	500	250,000	$6.114x10^6$	$6.197x10^6$	$1.210x10^5$

10	FSS	500	500	250,000	$6.733x10^6$	$7.174x10^6$	$9.775x10^4$
11	CFSS	500	500	250,000	$6.587x10^6$	$7.095x10^6$	$1.581x10^5$

Table 7.3.Performance of the Proposed Algorithms on Balerma Network

Index	Algorithm	Population size	NO. of iterations	Number of fitness evaluation	Best cost	Average best cost	Standard deviation
1	SPSO	1000	1000	1,000,000	$6.299x10^6$	$6.7290x10^6$	$5.542x10^6$
2	PE-PSO	--	--	--	--	--	--
3	ISEDPSO	1000	1000	1,500,000	$1.9334x10^6$	$1.9772x10^6$	$5.555x10^4$
4	PEDPSO	1000	1000	1,000,000	$1.9214x10^6$	$1.9421x10^6$	$3.936x10^4$
5	PSO-EO	1000	1000	1,022,700	$6.0460x10^6$	$6.7341x10^6$	$6.543x10^5$
6	OS-PSO	1000	1000	1,000,000	$2.2560x10^6$	$2.3250x10^6$	$8.958x10^4$
7	Shuffle-PSO	1000	1000	1,000,000	$4.1256x10^6$	$4.2551x10^6$	$1.716x10^5$
8	EMPSO-U	1000	1000	1,000,000	$3.58498x10^6$	$4.3070x10^6$	$5.195x10^5$
9	EMPSO-M	1000	1000	1,000,000	$3.74591x10^6$	$5.9315x10^6$	$1.213x10^6$
10	FSS	--	--	--	--	--	--
11	CAFSS	--	--	--	--	--	--

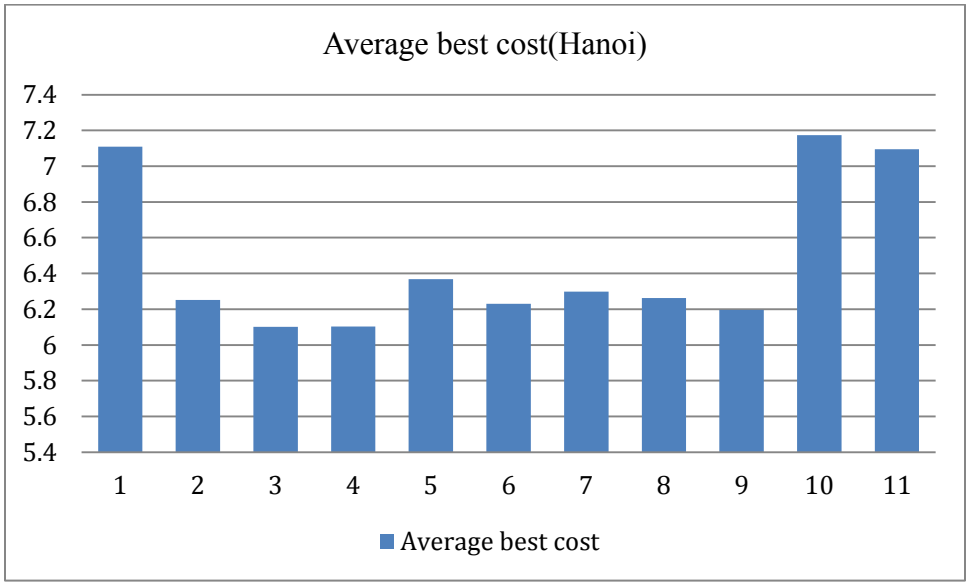


Figure 7.1.Average performance of the proposed algorithms on Hanoi network

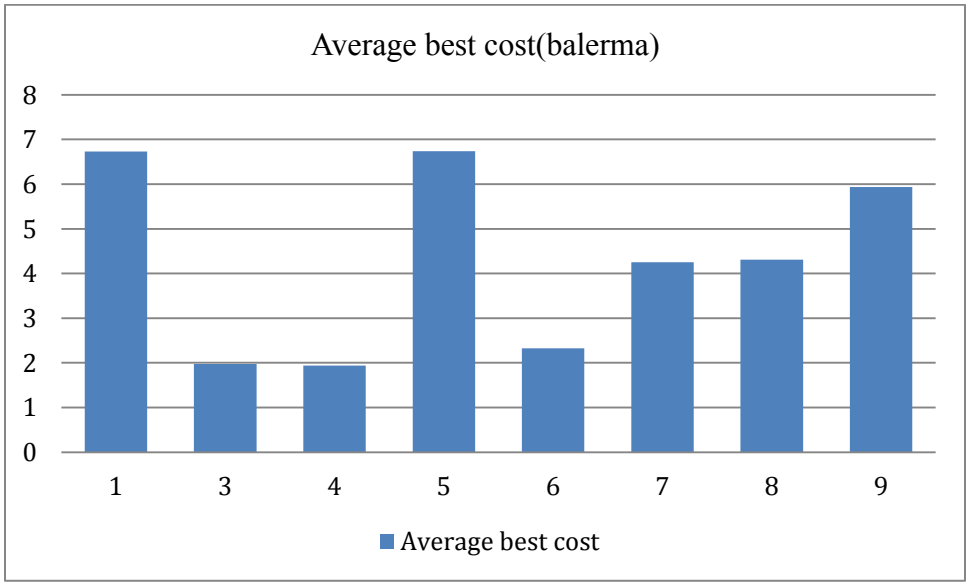


Figure 7.2.Average performance of the proposed algorithms on Balerma network

Table 7.4.Algorithms that Perform well on Both two Networks

Network	1st	2nd	3rd	4th
Hanoi	**ISEDPSO**	**PEDPSO**	EMPSO-M	**OS-PSO**
Balerma	**PEDPSO**	**ISEDPSO**	**OS-PSO**	Shuffle-PSO

7.4 Compare with Previous Work

We have only compared the proposed algorithms with each other so far, now we need to compare these algorithms with other peer method in previous work. Table 7.5 and 7.6 show the comparisons with some of the previous work. In order to compare the efficiency, the average number of fitness evaluations before it converges of the runs which successfully achieves the best cost. As we can see in Table 7.5, on Hanoi network, five proposed algorithms successfully get the best know cost in the literature and most of them are better than previous work in terms of the success rate. While for Balerma network, due to its high dimensionality and difficulty, there are only 3 algorithms successfully obtained the comparable best cost with previous work, and two of them break the record of minimal cost of Balerma network.

Table 7.5.Comparison of the Best cost Achieved by Different Algorithms (Hanoi)

Algorithms	Min cost	Average	Success rate	Min # of evaluations
GA([32] 2006)	6.173	n/a	n/a	26,457
ACO([57].2006)	6.134	n/a	n/a	35,433
HS([58], 2006)	6.081	n/a	1/81	27,721
PSHS([59],2009)	6.081	n/a	1/81	17,980
GHEST([39], 2010)	6.081	n/a	6/10	16,600
PSO ([35],2008)	6.133	n/a	3/100	n/a
HD-DDS([60], 2009)	6.081	6.252	8/100	5000,000
SAPSO([38],2010)	6.081	n/a	1/100	n/a
NLP-DE([61],2011)	6.081	n/a	98/100	48,724
GA([32] 2006)	6.173	n/a	n/a	26,457
ACO([57].2006)	6.134	n/a	n/a	35,433
ISEDPSO (this work)	**6.081**	6.102	**28/30**	**17,600**
PEDPSO (this work)	**6.081**	6.103	**27/30**	**23,400**
Shuffle-PSO (this work)	**6.081**	6.299	**11/30**	**126,500**
OS-PSO (this work)	**6.081**	6.231	**13/30**	**48,500**
PE-PSO (this work)	**6.081**	6.252	**8/30**	**20,000**

(Hazen-Williams roughness coefficient for hydraulic calculations is 10.6668. Price unit: M$)

Table 7.6.Comparison of the Minimal Cost Achieved by Different Algorithms (Balerma)

Algorithms	Min cost	Average	Success rate	Min # of evaluations
GA([32], 2006)	2.302,000	n/a	n/a	10,000,000
HS([58], 2006b)	2.018,000	n/a	1/81	10,000,000
HD-DDS([60], 2009)	1.940,923	2.165,000	n/a	30,000,000
GHEST([39], 2010)	2.002,387	n/a	1/10	290,500

NLP-DE([61], 2011)	1.923,000	1.927,000	n/a	1427,850
ISEDPSO(this work)	**1.933,407**	**1.976.672**	**16/30**	**89,700**
PEDPSO(this work)	**1.921.428**	**1.942.231**	**17/30**	**217,400**
OS-PSO(this work)	**2.256,000**	**2.3250,000**	**89,580**	**193,500**

(Hazen-Williams roughness coefficient for hydraulic calculations is 10.6668. Price unit: M€)

CHAPTER 8

8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this thesis, 11 algorithms are proposed for WDN optimization problem in total. Except conventional integer PSO (SPSO), all other 10 algorithms are the first time applied to this engineering problem. And also except SPSO and FSS, there are 5 algorithms(PEPSO, ISEDPSO, PEDPSO, OSPSO and CAFSS), which are brand new variants based on PSO and FSS, to our best knowledge, are also the first time been proposed.

Based on the elaborate analysis and intensive comparison in Chapter 7, following conclusions can be drawn:

1. For Hanoi network, all the proposed PSO variants outperforms the conventional integer PSO, which proves that all these adopted Machine Learning techniques successfully improve the search ability of SPSO on Hanoi network.
2. For Balerma network, due to its high dimensionality, the results are different. Except PSO-EO, all other 7 algorithms could outperform SPSO.

3. For Fish School Search, its overall performance is even worse than SPSO. Although its only variant CAFSS dose better than standard FSS, its performance still cannot be comparable to PSO and PSO based variants. So we can say PSO is much more suitable for this specific engineering optimization problem than FSS.
4. The ISEDPSO, PEDPSO and OSPSO are the best 3 algorithms for the problem of both small and large networks. It is notable that all these 3 algorithms include the same process, which learns from the historical good positions. This learning process is implemented by EDA process in practical application.
5. Comparing to the previous work in literature, there are 3 proposed algorithms are competitive, but more experiments still need to be carried out to further dig out the potential of these 3 algorithms.
6. For each specific proposed variant, we also have specific conclusions about the performance of that specific algorithm, which are given in previous corresponding chapters.

8.2 Future Work

Apart from the specific discussion of future work for each of the proposed algorithm, which have been already been given in previous corresponding chapters, there are some potential research directions from the overall point of view:

1. Optimization of Water Distribution Network is still a challenge, especially for large size network. It is still worthy to propose more alternative optimization methods for this problem.

2. Introduce more swarm intelligence algorithms to solve this engineering optimization problem.

3. From Machine Learning point of view, more learning methods could be tested to improve the performance of PSO and other swarm intelligence algorithms.

9 REFERENCES

[1] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.

[2] T. Back, M. Emmerich, and O. M. Shir, Evolutionary algorithms for real world applications, In *IEEE Comput. Intell. Mag.*, vol. 3, no. 1, pp. 64–67, Jan. 2008.

[3] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*, 1st ed. San Mateo, CA: Morgan Kaufmann, 2001.

[4] Schaake, J., and Lai, D., Linear programming and dynamic programming applications to water distribution network design. Rep. No. 116, Dept. of Civ. Engrg., Massachusetts Institute of Technology, Cambridge, Mass. (1969).

[5] Alperovits, E., and Shamir, U., Design of optimal water distribution systems. Water Resour. Res., 13(6), 885–900. (1977).

[6] Quindry, G. E., Brill, E. D., and Liebman, J. C., Optimization of looped water distribution

systems. J. Envir. Engrg., ASCE, 107(4), 664–679. (1981).

[7] Goulter, I. C., and Morgan, D. R., An integrated approach to the layout and design of water distribution networks. Civ. Engrg. Sys. 2(2), 104–113. (1985).

[8] Walski, T. M., State-of-the-art: Pipe network optimization. Computer applications in water resources, H. C. Toeno, ed., ASCE, New York, 559–568. 1985.

[9] Fujiwara, O., Jenchaimahakoon, B., and Edirisinghe, N. C. P., A modified linear programming gradient method for optimal design of looped water distribution networks. Water Resour. Res., 23(6), 977– 982. 1987.

[10] Kessler, A., and Shamir, U., Analysis of the linear programming gradient method for optimal design of water supply networks.'' Water Resource. Res., 25(7), 1469–1480. (1989).

[11] Lansey, K. E., and Mays, L. W., Optimization model for design of water distribution systems. Reliability analysis of water distribution systems, L. R. Mays, ed., ASCE, Reston, Va. (1989).

[12]Goldberg, D. E., and Kuo, C. H., (1987). Genetic algorithms in pipeline optimization. J. Comp. in Civ. Engrg., ASCE, 1(2), 128–141

[13] Dandy, G.C., Simpson, A.R. and Murphy, L.J., (1996). An improved genetic algorithm for

pipe network optimization", Water Resour. Res., 32, pp 449-458

[14] Montesinos P, Garcia-Guzman A, Ayuso JL., (1999). Water distribution network optimization using modified genetic algorithm. Water Resour Res 35(11):3467–3473

[15] Vairavamoorthy K, Ali M.,(2000). Optimal design of water distribution systems using genetic algorithms. Comput-Aided Civil Infrastruct Eng 15(5):374–382

[16] Morley M.S., Atkinson R.M., Walters G.A., (2001). GAnet: genetic algorithm platform for pipe network optimization. Advances in Engineering Software 32:467-475

[17] Edward Keedwell, Soon-Thiam khu., (2005). A hybrid genetic algorithm for the design of water distribution networks. Engineering Application of Artificial Intelligence 18:461-472

[18] Reca, J., and J. Martinez., (2006). Genetic algorithms for the design of looped irrigation water distribution networks, Water Resour. Res., 42, W05416, doi:10.1029/2005WR004383

[19] Kadu MS, Gupta R, Bhave PR., (2008). Optimal design of water networks using a modified genetic algorithm with reduction in search space. J Water Resour Plan Manage;134(2):147–60.

[20] Andrea Bolognesi, Cristiana Bragalli, Angela Marchi, Sandro Artina,(2010). Genetic Heritage Evolution by Stochastic Transmission in the optimal design of water distribution networks. Advances in Engineering Software 41:792-801.

[21] Milan Cristy., (2010).Hybrid Genetic Algorithm and Linear Programming Method for Least-Cost Design of Water Distribution Systems. Water Resour Manage 24, 1-24

[22] Idel Montalvo, Joaquin Izquierdo, Rafael Perez-Garcia, Michael M. Tung, (2008). Particle Swarm Optimization applied to the design of water supply systems. Computers and Mathematics with Applications 56, 769-776

[23] Idel Montalvo,Joaquı´n Izquierdo n, RafaelPe´ rez-Garcı´a, ManuelHerrera, 2010. Improved performance of PSO with self-adaptive parameters for computing the optimal design of Water Supply Systems. Engineering Applications of Artificial Intelligence. 23, 727-735.

[24] Maria da Conceica cunha and Joaquim Sousa, (1999). Water Distribution Network Design Optimization: Simulated Annealing Approach. Journal of Water Resource Planning and Management. 125(4), 215-221

[25] Maria da Conceicao Cunha, Luisa Ribeiro, Tabu Search Algorithms for Water Network Optimization, European Journal of Operational Research, 157(2004)746-758.

[26] Geem, Z.W., Kim, J.H. and Yoon, Y.N.,(2000). Optimal layout of pipe networks using harmony search, in Proc.of 4th Int. Conf. on Hydro-Science and Engineering,
Seoul, South Korea.

[27] Eusuff MM, Lansey KE. Optimization of water distribution network design using the shuffled frog leaping algorithm. J Water Resour Plan Manage 2003;129(3):210–25.

[28] S. Mohan and K. S. Jinesh Babu. Optimal Water Distribution Network Design with Honey Bee Mating Optimization, Journal of Computing in Civil Engineering, 2010: 24(1): 117-126

[29] Chien-Wei Chu, Min-Der Lin, Gee-Fon Liu, Yung-Hsing Sung. Application of Immune Algorithms on Solving minimum-cost problem of Water Distribution Network. 2008:48:1888-1900.

[30] Shie-Yui Ling and Md. Atiquzzaman, Optimal Design of Water Distribution Network Using Shuffled Complex Evolution, In Journal of Institution of Engineers, Singapore 44(1) 2004 pp 93-107.

[31] Aaron C. Zecchin, angus R. Simpson, Holger R. Maier, Michael Leonard, Andrew J. Roberts, Matthew J. Berrisford. 2006. Application of two ant colony optimisation to water distribution system optimisation. Mathematical and Computer Modelling 44 (2006) pp.451-468.

[32] Andrea Bolognesi, Cristiana Bragalli, Angela Marchi, Sandro Artina, Genetic Heritage Evolution by Stochastic Transmission in the optimal design of water distribution networks. Advances in Engineering Software 41:792-801. (2010).

[33] R. Banos C. Gil, J.Reca, G.G., Montoya A memetic algorithm applied to the design of water distribution networks. Applied Soft Computing. 10261-266,(2010).

[34] Fujiwara, O., and Khang, D. B., Correction to a two-phase decomposition method for optimal design of looped water distribution networks. Water Resour. Res., 27(5), 985–986. (1991).

[35] Reca, J., and J. Martinez., Genetic algorithms for the design of looped irrigation water distribution networks, Water Resour. Res., 42, W05416, doi:10.1029/2005WR004383. (2006).

[36]J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, IEEE. Service Center, Piscataway, NJ. (1995).

[37] R. Chiong (Ed.), Nature-Inspired Algorithms for Optimisation, Berlin: Springer-Verlag, April 2009. ISBN 978-3-642-00266-3.

[38] BASTOS FILHO, Carmelo J. A. ; LIMA NETO, Fernando B. de; LINS, Anthony J. C. C.; NASCIMENTO, Antônio I. S.; LIMA, Marília P., A Novel Search Algorithm based on Fish School Behaviour. In: 2008 IEEE International Conference on Systems, Man, and Cybernetics - SMC 2008, 2008, Cingapura.

[42] Min-Rong Chen, Xia Li, Xi Zhang, Yong-Zai Lu, (2010). A novel particle swarm optimizer hybridized with extreme optimization. vol.10, no.2, Pages 367-373.

[43] J. Kennedy, (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In Proc.Congr.Evol.Comput.pp.1931-1938.

[44] J. Kennedy and R. Mendes, “Population structure and particle swarm performance” in Proc. Congr. Evol. Comput., 2002, pp. 1671-1676.

[45] X.Li, 2010. Niching without niching parameters: Particle Swarm Optimization using a ring topology. IEEE Trans.Evol. Comput., vol. 14, no. 1, pp. 150-169. Feb. 2010.

[46] T. Krink, J. Vesterstrom, J. Riget. “Particle Swarm Optimization with Spatial Particle Extension”. Proceedings of the congress on Evolutionary Computation 2002.

[47] C. K. Monson and Kevin D. Seppi. Adaptive diversity in PSO. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06). ACM, New York, NY, USA, 59-66. 2006.

[48] P. Angeline, “Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences” in Proc. 7th Conf. Evol. Program., 1998, pp. 601-610.

[49] R.V. Kulkarni,(2007). Venayagamoorthy, G.K.; , "An Estimation of Distribution Improved Particle Swarm Optimization Algorithm," Intelligent Sensors, Sensor Networks and Information. ISSNIP 2007. 3rd International Conference on , vol., no., pp.539-544, 3-6 Dec.

[50] M. El-Abd and M.S. Kamel, “Particle Swarm Optimization with varying bounds” in IEEE congress on Evolutionary Computation.2007, pp. 4757-4761.

[51] C. K. Monson and Kevin D. Seppi. (2006). Adaptive diversity in PSO. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06). ACM, New York, NY, USA, 59-66.

[52] Y. Zhou and J.Jin (2006). EDA-PSO-a new hybrid intelligent optimization algorithm, in Proc. Of the Michigan University Graduate Student Symposium 2006.

[53] Mudassar Iqbal and Marco A. Montes de Oca(2006) M. Iqbal and M. A. Montes de Oca, “A estimation of distribution particle swarm optimization algorithm” in Proc. Of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence.

[54] R.V. Kulkarni,(2007). Venayagamoorthy, G.K.; , "An Estimation of Distribution Improved Particle Swarm Optimization Algorithm," Intelligent Sensors, Sensor Networks and Information. ISSNIP 2007. 3rd International Conference on , vol., no., pp.539-544, 3-6 Dec.

[55] M. El-Abd and M.S. Kamel, “Particle Swarm Optimization with varying bounds” in IEEE congress on Evolutionary Computation.2007, pp. 4757-4761.

[56] M. El-Abd, (2009). "Preventing premature convergence in a PSO and EDA hybrid,"

Evolutionary Computation. CEC '09. IEEE Congress on, vol., no., pp.3060-3066, 18-21.

[57] Hongcheng Liu, Liang Gao, Quanke Pan, (2011), A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. Expert System with Applications 38(2011)4348-4360.

[58] Chang Wook Ahn, Jinung An, Jae-Chern Yoo, (2012), Estimation of particle swarm algorithms: Combining the benefits of PSO and EDAs. 192(2012) 109-119.

[59] Feifei Zheng, Angus R. Simpson, and Aaron C. Zecchin. (2011). A combined NLP-differential evolution algorithm approach for the optimization of looped water distribution systems. Water Resources Research, Vol. 47, http://dx.doi.org/10.1029/2011WR010394. 2011.

[60] J. J. Liang and P. N. Suganthan, "Adaptive comprehensive learning particle swarm optimizer with history learning," in *Proc. Int. Conf. Simulated Evolution and Learning*, 2006, pp. 213–220.

[61] M. J. Zhang, X. Y. Gao, and W. J. Lou, "A new crossover operator in genetic programming for object classification," *IEEE Trans. Syst., Man, and Cybern. B.*,
vol. 37, no. 5, pp. 1332–1343, Oct. 2007.

[62] M. Li and H. Y. Tam, "Hybrid evolutionary search method based on clusters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 8, pp. 786–799, Aug.

2001.

[63] J. Kennedy, “Stereotyping: Improving particle swarm performance with cluster analysis,” in *Proc. Congr. Evolutionary Computation*, 2000, pp. 1507–1512.

[64] Y. J. Wang, J. S. Zhang, and G. Y. Zhang, “A dynamic clustering based differential evolution algorithm for global optimization,” *Eur. J. Oper. Res.*, vol. 183, no.
1, pp. 56–73, Nov. 2007.

[65] McLachlan, G., ed.: Mixture Models. Marcel Dekker, New York, NY (1988).

[66] Dempster, A., Laird, N., Rubin, D.: Maximum Likelihood from Incomplete Data via the EMAlgorithm. Journal of the Royal Statistical Society 39(1) (1977) 1–38

[67] Eberhart R.C., Kennedy J.. 1995. A new optimizer using particle swarm theory. In: 6th International Symposium on

[68] Arumugam M.S., Rao M.V.C.. 2008. On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems. Applied Soft Computing. 8 (1), 324–336.

[69] Ratnaweera A., Halgamuge S.K., Watson H.C.. 2004.Self-organizing hierarchical particle

swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation 8 (3), 240–255.

[70] Liao C.J., Tseng C.T., Luarn P.. 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. Computers and Operations Research. 34 (10), 3099–3111.

[71] G. Kendall, E. Soubeiga, and P. Cowling, “Choice function and random hyper heuristics,” in Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, SEAL. Springer, 2002, pp. 667– 671.

[72] J. E. Smith and T. C. Fogarty, “Operator and parameter adaptation in genetic algorithms,” Soft Computing - A Fusion of Foundations, Methodologies and Applications, vol. 1, pp. 81–87, 1997.

[73] J. E. Smith, “Credit assignment in adaptive memetic algorithms,” in Proceedings of the 9th annual conference on Genetic and evolutionary computation, ser. GECCO ’07. New York, NY, USA: ACM, 2007, pp. 1412–1419.

[74] D. Thierens, “An adaptive pursuit strategy for allocating operator probabilities,” in Proceedings of the 2005 conference on Genetic and evolutionary computation, ser. GECCO’05. New York, NY, USA: ACM, 2005, pp. 1539–1546.

[75] D. Thierens, “Adaptive strategies for operator allocation,” in Parameter Setting in Evolutionary Algorithms, ser. Studies in Computational Intelligence, F. Lobo, C. Lima, and Z. Michalewicz, Eds., vol. 54. Springer Berlin / Heidelberg, 2007, pp. 77–90.

[76] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, “Adaptive operator selection with dynamic multi-armed bandits,” in Proceedings of the 10th annual conference on Genetic and evolutionary computation. GECCO ’08. New York, NY, USA: ACM, 2008, pp. 913–920.

[77] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag, “Extreme compass and dynamic multi-armed bandits for adaptive operator selection,” in Evolutionary Computation, 2009. CEC ’09. IEEE Congress on, may 2009, pp. 365 –372.

APPENDIX

SOURCE CODE

All the experiments for this thesis project are carried on Matlab2012 platform (Basic version with one Global Optimization Tool Box and Statistics Tool Box). The source code is not released at this time, if you're interested, please contact the author at qixuewei@uga.edu .