

**NEW INNOVATIONS IN SWARM OPTIMIZATION VIA MULTI-CRITERION  
DECISION MAKING**

By

Ahmed I. EL-Gallad

Submitted  
in partial fulfillment of the requirements  
for the degree of

**DOCTOR OF PHILOSOPHY**

Major subject: Electrical and Computer Engineering Department

at

**DALHOUSIE UNIVERSITY**

Halifax, Nova Scotia

August, 2007

© Copyright by Ahmed EL-Gallad, 2007



Library and  
Archives Canada

Bibliothèque et  
Archives Canada

Published Heritage  
Branch

Direction du  
Patrimoine de l'édition

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file    Votre référence*

*ISBN: 978-0-494-35792-7*

*Our file    Notre référence*

*ISBN: 978-0-494-35792-7*

#### NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

#### AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

  
**Canada**

DALHOUSIE UNIVERSITY

To comply with the Canadian Privacy Act the National Library of Canada has requested that the following pages be removed from this copy of the thesis:

Preliminary Pages

Examiners Signature Page

Dalhousie Library Copyright Agreement

Appendices

Copyright Releases (if applicable)

*I dedicate this work  
To my mother, my wife and my sons: Fares and Joseph  
In memory of my beloved father!*

## Table of Contents

<b>List of Tables</b>	ix
<b>List of Figures</b>	xi
<b>List of Symbols and Abbreviations</b>	xiv
<b>Acknowledgments</b>	xvii
<b>Abstract</b>	xviii
<b>1 Introduction</b> .....	1
1.1 Motivation.....	1
1.2 Darwin evolution vs. animal aggregations and swarms.....	3
1.3 Proximity to autonomous agents.....	5
1.4 Objectives .....	5
1.5 Contributions.....	6
1.6 Methodology .....	7
1.7 Thesis outlines .....	7
<b>2 Literature Review</b> .....	9
2.1 Introduction.....	9
2.2 Cellular automata, emergence and swarms.....	10
2.3 Original particle swarm optimizer .....	13
2.4 Inertia weight .....	16
2.5 Analysis of the particle swarm.....	19
2.5.1 Trajectory analysis .....	20
2.5.2 Stability analysis .....	22
2.5.3 Stability of the stochastic PSO.....	29
2.5.4 Local and global convergence .....	31
2.6 Diversity and PSO.....	33
<b>3 New Formulation for the Swarm Optimization</b> .....	37
3.1 Introduction.....	37
3.2 Complex adaptive system .....	38

3.2.1 Adaptability vs. determinability.....	38
3.2.2 Complexity vs. complicatedness.....	39
3.3 Understanding swarms and animal aggregations.....	39
3.3.1 Gregariousness.....	39
3.3.2 Social tolerance.....	40
3.3.3 Social forces and swarm's dynamism.....	41
3.4 Free willing in animal world.....	41
3.5 Proposed Basic behaviors .....	42
3.5.1 Evidence of memory retrieval.....	44
3.5.2 Imitation.....	45
3.5.3 Play .....	45
3.6 Summary.....	46
<b>4 The Proposed Technique.....</b>	<b>47</b>
4.1 Introduction.....	47
4.2 Multi criterion decision making (MCDM) .....	48
4.2.1 Crisp reasoning based procedure .....	49
4.2.2 Fuzzy reasoning based procedure .....	50
4.3 Elements of decision making process.....	51
4.3.1 Goals .....	51
4.3.2 Alternatives.....	52
4.3.3 Criteria .....	52
4.3.3.1 Fitness .....	52
4.3.3.2 Levenshtien edit Distance.....	53
4.3.3.3 Criteria Fuzzification .....	55
4.3.4 Decision rule: fuzzy OWA aggregation operator .....	57
4.3.5 Agents' position update .....	59
<b>5 Applications .....</b>	<b>62</b>
5.1 Introduction.....	62
5.2 The Traveling Salesman Problem.....	62
5.2.1 Background.....	63
5.2.2 TSP and graph representation .....	64

5.2.3 Demonstrative example .....	66
5.2.4 Decision alternatives and criteria.....	68
5.2.5 Decision rule (OWA).....	69
5.2.6 Larger scale TSP problems .....	74
5.3 Quadratic Assignment Problem .....	76
5.3.1 The assignment problem.....	79
5.3.2 Linear assignment (LAP).....	79
5.3.3 The quadratic assignment problem (QAP) .....	80
5.3.4 QAP benchmarks .....	81
5.3.5 Testing and results .....	83
5.4 Significance of error differences.....	93
5.4.1 Anderson-Darling test of normality .....	93
5.4.2 Mann-Whitney test.....	94
5.4.3 U-statistic .....	95
5.4.4 Performing the test.....	96
5.5 MCDM-PSO versus Traditional PSO .....	100
5.6 Concluding remarks .....	103
<b>6 Extension to Continuous Variables Optimization .....</b>	<b>105</b>
6.1 Introduction.....	105
6.2 The MCDM-PSO model.....	105
6.2.1 Momentum influence .....	106
6.2.2 Memory retrieval influence.....	106
6.2.3 Imitation influence .....	107
6.2.4 Play influence.....	108
6.2.5 Position update equation.....	109
6.3 Computational experiments .....	109
6.3.1 Benchmark problems .....	110
6.3.2 Test environment and MCDM-PSO setting.....	126
6.3.3 Algorithms used for comparison.....	127
6.4 Test results .....	128
6.4.1 MCDM-PSO vs. traditional PSO.....	128

6.4.2 MCDM-PSO vs. other algorithms .....	138
6.5 Summary .....	141
<b>7 Conclusions and future work</b> .....	142
7.1 Summary .....	142
7.2 Future work .....	145
Bibliography .....	146
APPENDIX .....	161



## List of Tables

Table 2.1: Transition function using Rule 30 for one-dimensional cellular automaton. ..	11
Table 4.1: Decision table [75].....	49
Table 4.2: Calculation of LD between permutations $t$ and $s$ .....	54
Table 4.3: Limits on the fitness and the distance for fuzzification process.....	55
Table 4.4: Alternatives vs. criteria.....	60
Table 4.5: Ranked fuzzyfied ( $RF$ ) values for each alternative .....	60
Table 5.1: Decision attributes for agent $X$ at iteration $k$ .....	68
Table 5.2: Fuzzification Limits.....	68
Table 5.3: Ranked fuzzy measures .....	69
Table 5.4: Five different decision strategies [81] .....	70
Table 5.5: Simulation results for different TSPs .....	75
Table 5.6: Solving Nug30.....	78
Table 5.7: Results for Nug-type problems.....	86
Table 5.8: Results for Bur-type problems.....	86
Table 5.9: Results for Chr-type problems.....	87
Table 5.10: Results for Esc-type and Els-types problems .....	87
Table 5.11: Results for Had-type and Kra-type problems .....	88
Table 5.12: Results for Scr-type and Rou-type problems.....	88
Table 5.13: Results for Tho-type, Wil-type and Ste-type problems.....	88
Table 5.14: Results for Sko-type problems.....	89
Table 5.15: Results for Tai-a and Tai-c type problems .....	89
Table 5.16: Results for Tai-b type problems .....	90
Table 5.17: Results for Lipa-a type problems.....	90
Table 5.18: Results for Lipa-b type problems .....	90
Table 5.19: Ranks for average case samples .....	99
Table 5.20: Test Statistics for average case samples .....	99
Table 5.21 Ranks for best case samples.....	99
Table 5.22: Test Statistics for best case samples .....	99

Table 5.23: Values of parameters used in case of PSO and ACO [134] .....	100
Table 5.24: MCDM-PSO versus ant colony (ACO) and traditional PSO .....	101
Table 6.1: key players in memory retrieval relation. ....	106
Table 6.2: key players for imitation influence. ....	107
Table 6.3: Both sides of the play influence.....	109
Table 6.4: Results for problems $f_1$ to $f_{10}$ on 30 dimensions. ....	129
Table 6.5: Results for problems $f_1$ to $f_{10}$ on 100 dimensions .....	130
Table 6.6: Results for problems less than 30 dimensions. ....	131
Table 6.7: Results for MCDM-PSO, ARPSO and SEA on problems of 30 dimensions or less. .....	139
Table 6.8: Results for MCDM-PSO, ARPSO and SEA on problems of 100 dimensions.	140
Table 6.9: MCDM-PSO vs. some PSO variants on problems of 30 dimensions. ....	141

## List of Figures

Figure 2.1: Visualizing Rule 30 in black and white [26].....	12
Figure 2.2: The evolved pattern after 15 steps [26].....	12
Figure 2.3: PSO publications from 1995-2006 (IEEE).....	16
Figure 3.1: Combining basic behaviors to form a candidate move .....	44
Figure 4.1: Membership function used in calculating fuzzy sets of the criteria.....	56
Figure 4.2: The OWA in the perspective of risk-tradeoff space of fuzzy aggregation operators .....	58
Figure 4.3: OWA-supported decision making [5] .....	59
Figure 5.1: Symmetrical and Asymmetrical TSP representation.....	65
Figure 5.2: Attributes of a captured instant $k$ for 17-cities ASTP problem. ....	67
Figure 5.3: Recursive displacement of items among permutations. ....	72
Figure 5.4: the pseudo code for the main steps of the proposed technique .....	74
Figure 5.5: Comparison between the traditional PSO and the proposed technique.....	76
Figure 5.6: Permutation encoding for LAP of $n=10$ . ....	80
Figure 5.7: Classification of QAP benchmarks [129].....	83
Figure 5.8: Differences between gaps assuming normality (average case).....	92
Figure 5.9: Differences between gaps assuming normality (best case). ....	92
Figure 5.10: Anderson-Darling test results.....	94
Figure 5.11: Cumulative distribution function.....	95
Figure 5.12: Nonparametric distribution fitting (average case).....	98
Figure 5.13: Nonparametric distribution fitting (best case).....	98
Figure 5.14: Probability Plot of Levenshtien distance matrix for traditional PSO after 100 iterations.....	102
Figure 5.15: Probability Plot of Levenshtien distance matrix for MCDM-PSO after 100 iterations.....	103
Figure 6.1: Visualization of the Sphere model .....	111

Figure 6.2: Visualization of the Rotated hyper-ellipsoids function.....	111
Figure 6.3: Axis parallel hyper-ellipsoids.....	112
Figure 6.4: Visualization of the Rastrigin function; (a) between -5 and 5 and (b) between -1 and 1.....	113
Figure 6.5: Visualization of the Schwefel function .....	114
Figure 6.6: Visualization of the Rosenbrock function; (a) between -2 and 2 and (b) around the local minimum (between -0.5 and 1.5) .....	114
Figure 6.7: Visualization of the Griewangk function; (a) in the full range, (b) between -50 and 50 and (c) between -8 and 8 .....	116
Figure 6.8: Visualization of Ackley path function; (a) between -30 and 30 and (b) between -2 to 2 .....	116
Figure 6.9: Visualization of the sum of different powers function.....	117
Figure 6.10: Visualization of the Zakharov function.....	117
Figure 6.11: Visualization of the Langerman function; (a) for the first and second variables (b) for the second and third variables with first variable set to zero .....	119
Figure 6.12: Visualization of the Michalewicz function; (a) with variable 1 and variable 2 in the range 0 to 3, (b) in the range 1.5 to 2.5 and (c) with variables 3 and 4 while variables 1 and 2 are set to zero in the rang 0 to 3 .....	119
Figure 6.13: Visualization of the Easom function; (a) between -20 and 20 and (b) between 1 and 5.....	121
Figure 6.14: Visualization of the Six-hump camel back function; (a) wide overview (b) focused view around the minima .....	121
Figure 6.15: Visualization of the Branin function .....	122
Figure 6.16: Visualization of the Hump function .....	122
Figure 6.17: Visualization of the Matyas function .....	124
Figure 6.18: Visualization of the Shubert function.....	124
Figure 6.19: Visualization of the Trid function .....	125
Figure 6.20: Convergence of the 30 dimensions Sphere function. ....	132
Figure 6.21: Convergence of the 30 dimensions Rotated hyper-ellipsoids function.....	132
Figure 6.22: Convergence of the 30 dimensions Axis parallel hyper-ellipsoids function....	133
Figure 6.23: Convergence of the 30 dimensions Rastrigin function. ....	133

Figure 6.24: Convergence of the 30 dimensions Schwefel function. ....	134
Figure 6.25: Convergence of the 30 dimensions Rosenbrock function.....	134
Figure 6.26: Convergence of the 30 dimensions Griewangk function .....	135
Figure 6.27: Convergence of the 30 dimensions Ackley function.....	135
Figure 6.28: Convergence of the 30 dimensions The sum of different powers function.....	136
Figure 6.29: Convergence of the 30 dimensions Zakharov function.....	136
Figure 6.30: Convergence of the 10 dimensions Trid function. ....	137

## List of symbols and abbreviations

PSO	Particle swarm optimizer
WLC	Weighted linear combination
SAW	Simple additive weighting
MCDM	Multi-criterion decision making
OWA	Fuzzy ordered weighted average
TSP	Traveling salesmen problem
QAP	Quadratic assignment problem
SGA	Standard genetic algorithm
SEA	Simple evolutionary algorithm
$(X_k^i, Y_k^i)$	Position of particle $i$ at iteration $k$
$(X_k^g, Y_k^g)$	Global best position at iteration $k$
$(X^i, Y^i)$	Previous best position of particle $i$
$VX$	Particle's velocity in X direction
$VY$	Particle's velocity in Y direction
$c_1$ and $c_2$	System's parameters
$rand$	Uniformly random variable between 0 and 1
$f_k^i$	Fitness value for the $i^{th}$ particle at $k^{th}$ iteration
$S$	Swarm size
NFL	No free lunch theory
$W$	Inertia weight
$\chi$	Constriction factor
$E\bar{X}(t)$	Expectation of $\bar{X}(t)$
$ED(t)$	Variance of $\bar{X}(t)$
$\{g(k)\}$	Sequence of the generated global best positions
$R_{opt}$	Optimality region
ARPSO	Attractive-repulsive PSO
SOC	Self organized criticality

LD	Levenshtien edit distance
<i>gbestx</i>	Position due to global best fitness
<i>pbestx</i>	Position due to personal best fitness in agent memory
<i>Nbestx</i>	Position of agent that has the best fitness in the neighborhood
<i>Randx</i>	Randomly generated position
<i>gbest</i>	Global best fitness in the domain
<i>pbest</i>	Personal best fitness in agent memory
<i>Nbest</i>	Best fitness value in an agent's neighborhood
<i>Rndft</i>	Fitness due to random position
<i>Xft</i>	Fitness at current position $X^K$
<i>RF</i>	Ranked fuzzy measure
TSPLIB	Online library of the traveling salesman problem
QAPLIB	Online library of the quadratic assignment problem
STSP	Symmetrical traveling salesman problem
ATSP	Asymmetrical traveling salesman problem
ICEO	International Contest on Evolutionary Optimization
$\pi : A \leftrightarrow B$	bijective (one-to-one and onto) function from a set A to a set B
$RI_i$	Imitation ranking index
$RI_r$	Memory retrieval ranking index
$RI_m$	Momentum ranking index
$RI_p$	Play ranking index
LAP	Linear assignment problem
BKS	Best known solution
AVG	Average solution over given number of replications
AVG Gap	Difference between the best known and the average solution
Gap	Difference between the BKS and the best solution
$U$	Mann-Whitney test statistic
<i>cdf</i>	cumulative distribution function
P-value	Level of significance
ACO	Ant colony optimization

STD	Standard deviation
$\eta$	Momentum rate
UPSO	Unified particle swarm optimizer
FDR-PSO	Fitness-distance ratio based particle swarm optimizer
FIPS	Fully informed particle swarm optimizer
CPSO-H	Hybrid Cooperative particle swarm optimizer
CLPSO	Comprehensive learning particle swarm optimizer



# Acknowledgements

I would like to thank my thesis advisor Dr. Mo El-Hawary for his guidance and help. During the period of my study I learned a lot from him; not only in my research but also in several aspects of my life. It is really an honor to be one of Dr. El-Hawary's students.

I would like to thank the supervisory committee members: Dr. Timothy Little and Dr. William Phillips.

I thank Dr. Magdy Salama, the external examiner, for the time spent to read this work in great details and for his valuable suggestions.

Finally, I would like to express my gratitude to my wife: Ghada Gabr for her patience and endless support.

# Abstract

This thesis presents a new formulation for the swarm optimization technique as a system of autonomous agents. The proposed technique is based on an intimate understanding of swarms and animal aggregations in an attempt to simulate cognitive thinking of their members. The dynamic balance between gregarious and social intolerance behaviors demonstrated by social animals is used to form the swarm and keep its persistence. In this work, members of the swarm are represented by agents that enjoy a certain degree of freewill to respond adaptively to changes on the states of their swarm mates. Adaptive responses are reflected on the way agents move inside the problem domain. A new set of basic behaviors is defined, namely imitation, memory retrieval, momentum, and play. A multi-criterion decision making process (MCDM) is employed to update positions of swarm members in the problem space. Decision making alternatives are defined from the set of basic behaviors. Fitness and diversity characterize the decision criteria that are used to measure the performance of each alternative. Levenshtien edit distance is used to measure the distance between agents in the genotype space. Criteria are then standardized by means of fuzzy sets. Fuzzified values of criteria are aggregated by the fuzzy ordered weighted average (OWA) to reach a single evaluation function. The overall decision making process is made to promote both fitness and diversity. The proposed technique is tested using the traveling salesmen (TSP) and quadratic assignment problems (QAP). Results and comparisons show that the technique outperforms the traditional particle swarm optimizer (PSO). Also, a comparison of the proposed technique with the standard genetic algorithm (SGA) shows that comparable results can be obtained. An extension of the proposed technique is also proposed to solve optimization problems with continuous variables. For this class of optimization, a large set of diverse benchmark problems is used to test the proposed technique. A comparison of the performance with the simple evolutionary algorithm SEA and many other particle swarm variants is also carried out. Results show that the proposed technique outperforms other techniques included in the comparison in almost all the tested problems.

# Chapter 1: Introduction

## 1.1 Motivation

Animate matters do not behave the way inanimate complicated machines do. There is always some sort of unforeseen randomness associated with the behavior of animate matters, which is a feature that has always fascinated researchers observing natural phenomena. The development of sciences like artificial life (Alife), complex adaptive behaviors, and autonomous agents is a strong proof of that fascination. However, designing a fully autonomous machine is very unlikely due to the fact that consciousness and freewill cannot be coded in a computer language. Nevertheless, there are some successful attempts to “resemble” such behaviors in a digital world.

Craig Reynolds [1] pioneered this field when he introduced the idea of boids (bird + android =boid [2]). Via mere observation, he simulated the process of birds flocking. His model was built using a decentralized bottom-up principle where no central control is imposed and the state of each individual in the swarm depends only on the state of its adjacent individuals. Reynolds also defined the following steering rules for the boids:

- 1) Cohesion rule: to allow individuals to fly close to their flock mates.
- 2) Separation rule: to avoid collision with nearby flock mates.
- 3) Alignment rule: to match the direction with others.

In 1995, James Kennedy and Russell Eberhart [3] adopted Reynolds’s model to solve optimization problems. Specifically, they exploited the cohesion and the alignment

rules of Reynolds' and eliminated the separation rule [4] developing what is now known as the particle swarm optimizer (PSO). Instead of aimless flocking, they employed a simulated "cornfield vector." This vector measures how far an individual bird is from a food source, which is analogous to a function's optimal solution. Surprisingly, this simple paradigm is able to comparably solve optimization problems in several engineering applications. However, the PSO is still relatively new and there are some drawbacks and issues that need to be addressed further.

First of all, the PSO paradigm is based on Reynolds' model, which has been criticized due to the lack of ethological evidence and reliance on mere observation. Moreover, Reynolds himself stated that "the animations showing simulated flocks built from this model seem to correspond to the observer's intuitive notion of what constitutes flock-like motion. However, it is difficult to objectively measure how valid these simulations are." [1].

Second, the PSO system iterates by updating the so-called position equation. This equation is constituted by adding two subjectively weighted terms namely: a social component and a cognitive component. Aggregation of this type is called a weighted linear combination (WLC) (or simple additive weighting (SAW)). Although it is not a pure crisp application of AND/OR, reaching a decision on candidate moves using this rationale is highly questionable [5]. Moreover, employing crisp logic expressed by an accurate sense of geometrical terms like velocity and distance in swarms is hardly imaginable [6].

Third, like many other heuristics, the PSO suffers from the problem of the premature convergence. Members of the swarm could be dragged into the same region (or point) in the search space causing a complete stagnation. This is a serious problem and has raised the concern of many researchers in the field. Unfortunately, instead of addressing causes, most of the work related to this issue focused on techniques to escape local minima [7]. However, there are a few attempts that were suggested to directly reduce the chance of trapping in local optima [8-12]. Rui Mendes [7] attributed the premature convergence in the PSO to two main reasons:

1. Fully informed dense swarms: all members of the swarm have access to the global best solutions in the search space. Accordingly, these solutions are rapidly propagated through iterations causing a formation of clusters in certain regions.
2. Unused wealth of information: the PSO iterates by means of global best solutions and best solutions found by its individuals. So, any other solution that is less in quality than those two types are normally discarded as well as their corresponding positions, despite the fact that these solutions might have valuable information about other regions in the search space.

Discarding the separation rule defined by Reynolds [1] could also be another reason for the premature convergence. This rule ensures that the flock mates fly close, but not too close to each other in order to avoid collision. Collision in the particle swarm optimizer means that two or more members occupy the same position in the genotypic landscape. As a result, the diversity level needed to ensure that the system keeps iterating, drops and eventually premature convergence takes place.

Fourth, due to its simplicity, many PSO modifications and variations have been proposed over the last ten years [8-15]. Going through the literature one can easily discover that the concept of the PSO is not fully understood. While there are many proposals that have positively influenced the performance of the PSO, there are also a few that appear to have a mixed understanding of the PSO and techniques that are built after Darwin's evolution.

## 1.2 Darwin's evolution vs. animal aggregations and swarms

Swarm-based optimizers and evolution-based techniques have common features such as population search mechanisms and random initialization. However, there are essential differences between the two techniques. In this section, these differences are explained in greater detail.

*Simulated time frame:*

- Evolution-based techniques represent a process that takes place over the accumulation of millions of years.

- Swarm optimizers simulate short time missions such as migration, flocking, roosting, and group hunting.

*Principle:*

- Techniques built by Darwinian evolution adopt the principle of survival of the fittest.
- In swarm optimizers, a collective intelligence emerges by interactive collaboration.

*Members:*

- Members of the population in the case of Darwinian techniques die off and new offspring are produced through generations [13].
- Members of the swarm never die off and the same members continue to carry out the mission until the end.

*Persistence:*

- In evolution based algorithms, advances made in the search space are achieved by sexual reproduction (crossover) and asexual random variation (mutation).
- In swarms, advances are made by updating the positions of the swarm members.

*Sources of innovation:*

- In the case of Darwinian evolution, the main sources of innovation are genetic operators such as crossover. This is a quasi-oriented process to combine two parents in order to produce an offspring.
- In a traditional PSO, simulated cognitive and social components are subjectively weighted and added together to form members' candidate moves. In the proposed technique this step is replaced by a rational decision making process.

*Random variations:*

- Generally, random variations are used to elevate the level of population diversity that might be negatively affected due to the recombination process of two or more solutions (in the genotypic sense). In Darwinian algorithms, the mutation process is simulated to achieve this purpose.
- In a traditional PSO, factors like random velocity, turbulence [14], and craziness [15] are usually used as sources of innovation in the search space. In the proposed

technique, the play behavior demonstrated in the animal kingdom is simulated to add a more naturalistic look to the swarm.

### 1.3 Proximity to autonomous agents

Autonomy of agent-based systems is just a relative (not absolute) feature that is set by the designers. However, agents existing in a “digital” world have some properties that were originally borrowed from agents in the real world. These properties can be stated as follows [16,17]:

1. *Autonomy*: the power of decision making and taking control over their own actions.
2. *Identity and character*: agents should demonstrate a certain degree of personality to respond differently to others’ attitudes, to compare behaviors with those previously observed, and to explore by means of random actions.
3. *Sociability*: the ability to communicate with other agents.
4. *Flexibility*: actions are not explicitly coded.
5. *Adaptive learning*: agents should be able to change their behavior as a result of previous learning experiences.
6. *Imitative interaction*: agents should possess behaviors that allow imitation.
7. *No specific role*: no specific role is assigned to each agent.

The above properties summarize the main characteristics and the structure of autonomous agent systems that are used to design the proposed technique.

### 1.4 Objectives

The main goals of the thesis can be summarized as follows:

1. Reformulating the idea of swarm optimizers for better performance and to gain a clearer understanding in order to distinguish its identity among existing heuristic techniques.
2. Introducing an autonomous agent system for the swarm that behaves in a more realistic fashion due to its reliance on a credible study from the field of sociobiology.

3. The final intention of this work is to overcome most of the earlier mentioned drawbacks in the traditional particle swarm optimizer.

### **1.5 Contributions**

This work is highly motivated by the ability of animal aggregations to adapt to changes in their environment. The main intention is to overcome the previously discussed drawbacks in the traditional PSO. Accordingly, the contributions of this work can be summarized as follows:

1. Adopting a scientific ethological basis in forming and keeping the persistence of the proposed swarm, instead of simple behavior observations used in the traditional PSO.
2. Defining a new set of basic behaviors that allows proximity to autonomous agent systems.
3. Employing a rational decision making theory to obtain candidate moves in the problem space instead of the subjective weighted addition used in the case of traditional PSO.
4. Applying fuzzy logic to simulate the balance between gregariousness and social intolerance behaviors.
5. Incorporating the diversity factor in the preferences of swarm members along with the existing fitness factor.
6. Employing the Levenshtien edit distance [18] to measure the distance between two individuals in the genotype landscape.
7. Addressing one of the most important problems in the field of combinatorial optimization. The proposed optimizer is extensively tested using the quadratic assignment problem.
8. Extending the proposed technique to handle optimization problems with continuous variables. A diverse set of optimization problems is chosen from the literature to test the technique.



## **1.6 Methodology**

In order to achieve the goals set for this thesis, the following steps are carried out:

1. Reviewing materials on sociobiology to understand mechanisms of animal aggregations and related behaviors.
2. Studying the multi-criterion decision making theory and criteria aggregation functions.
3. Applying the acquired knowledge to design the optimizer.
4. Testing the algorithm using well known problems in the field of combinatorial optimization (traveling salesmen and quadratic assignment problems).
5. Testing the proposed technique using real-valued optimization problems.
6. Analyzing and comparing simulation results with those obtained by other techniques.
7. Reviewing and monitoring new related work through journals and conferences' participation.

## **1.7 Thesis outline**

Chapter 2 starts with an explanation of terminologies associated with the particle swarm optimizer followed by a description of the original version and different variations of the algorithm. Other related issues such as stability, convergence, and diversity are also reviewed.

Chapter 3 presents the necessary biological background needed for the theory used in building the proposed technique.

Chapter 4 presents the proposed technique along with a detailed description of new mechanisms employed to build the algorithm.

Chapter 5 is dedicated to testing the performance of the technique with the traveling salesmen and quadratic assignment problems. The comparisons and results are also presented in this chapter.

In chapter 6, the proposed technique is extended to handle optimization problems with continuous variables. Experimental tests of a large set of benchmark problems along with comparisons with other heuristic techniques are also included in this chapter.

Chapter 7 summarizes findings of this work and suggests some directions for future research.

## **Chapter 2: Literature Review**

### **2.1 Introduction**

Since the dawn of civilization, human beings have enormously influenced the way of life on Earth. More than any other living species, humans have developed major innovations and great achievements that would not be possible without intelligence. The intelligence that evolves by abstract reasoning is a unique feature to humans. Abstract reasoning is defined as “the ability to analyze information and solve problems on a complex intangible level.” [19] There is no doubt that humans are immensely proud of their intelligence in a way that highly affects their desire to design smart machines. Scientists have always dreamed of designing a machine that can perform tasks beyond computing. They have dreamed of machines that are able to learn, adapt, think, make decisions, and possess intelligence. It turns out that human intelligence is too complex to synthesize and experiences in this direction are very modest (neural networks, expert systems, and problem-oriented robotic schemes). Perhaps the ultimate artificial intelligence model would be a human clone.

Because intelligence is social, scientists started looking at other species that live in social groups. They found out that among animals, only those who live in aggregations and swarms are able to survive longer and adapt better than others. Cellular automata theory and the growing interest of decentralized thinking trends have highly contributed to the development of the artificial life science (Alife). It is the field of simulating behaviors of natural complex systems, either out of fascination or to better understand these behaviors.

Reynolds’ boids model [1] is considered a milestone in this field. He simulated the flocking behavior of birds using simple decentralized interactions among simulated birds. In 1995 Kennedy and Eberhart [3] were able to manipulate Reynolds’ model to

solve optimization problems developing a new technique called a particle swarm optimizer. In this chapter, a review of the literature on the particle swarm optimizer is given along with some variants of the algorithm.

## 2.2 Cellular automata, emergence, and swarms

Generally, the concept of automaton is very old and can be described as a mechanism that is able to self-operate by following a sequence of pre-described instructions without intimate intrusion of human element [20]. Von Neumann is a very famous mathematician who lived between 1903 to 1957 and who is known for his contributions in both sequential and parallel computing systems. His work aimed at a broader sense of automata, a self-replicating machine that is able to replicate itself and create automata that are even of a higher order than itself [21]. Von Neumann first tried to design a kinematic model for that automaton but soon discovered that it would be hard to work with it due to motion and geometry considerations. He later employed the notion of “cell space” as per a suggestion from his colleague mathematician Stanislaw Ulam to reduce the spatial complexity from three dimensions to two [22]. Cellular approaches were then adopted to describe self-replicating automata that are now known as cellular automata. A cellular automaton is a decentralized computing structure that facilitates the emergence of complex behaviors from simple local interactions among its cells. It consists of a regular array of cells on a grid which can be of  $n$  dimensions (typically one to three) [23]. Each cell on the grid can be in one of a finite number of states. The dynamism of the cellular automaton is a result of progression in a discrete time by updating the state of each cell using a set of pre-specified deterministic rules called a transition function. The state of each cell in the succeeding step is completely determined by its current state and the states of its neighboring cells. The one-dimensional structure is the simplest type of cellular automata. In this type the automata array is one row of cells, where each cell has one of two possible cases (for example; 1 or 0). The neighborhood is defined by 3 cells: the center, the left, and the right cell. The neighborhood possibilities are represented by binary numbers starting from  $(111)_2$  descending to  $(000)_2$  [24]. The transition function that determines the state of each cell in

the next step can be any form of the known updating rules. For instance, *Rule 30* is considered for this example. This rule is proposed by Stephen Wolfram [25] to map three bits to one bit according to the logic expression:

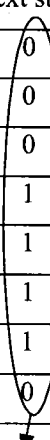
$$\text{Rule 30: } \text{XOR}[p, \text{OR}[q, r]] \quad (2-1)$$

The variables  $p$ ,  $q$ , and  $r$  are states of the left, center, and right cell respectively.

Accordingly, the rules can be arranged as in Table 2-1.

**Table 2.1: Transition function using Rule 30 for a one-dimensional cellular automaton.**

Left neighbor $p$	Current state $q$	Right neighbor $r$	Next state
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0


 $(00011110)_2 = (30)_{10}$

The rule is named by the result of evaluating the logic expression over the given neighborhood topologies in the specified order. In that case, the expression described by Equation (2-1) for the arrangement shown in Table 2-1 is equal to  $(00011110)_2$ , which corresponds to 30 in the decimal system. To visualize the complex pattern that has emerged by updating the state of each cell according to this rule, the color black is given to cells that have *1-state* and white for cells with *0-state* [25]. In that sense, Rule 30 can be described as shown in Figure 2-1. Starting with a row of cells with only one “life” (black) cell in the middle, the pattern shown in Figure 2-2 will emerge after 15 progressive steps of states’ update by using the specified rule.

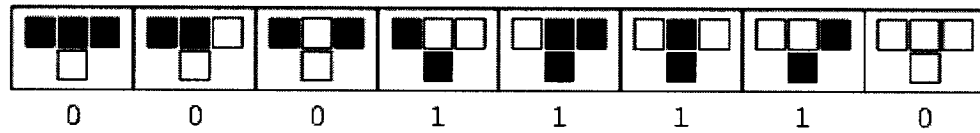


Figure 2.1: Visualizing Rule 30 in black and white [26]

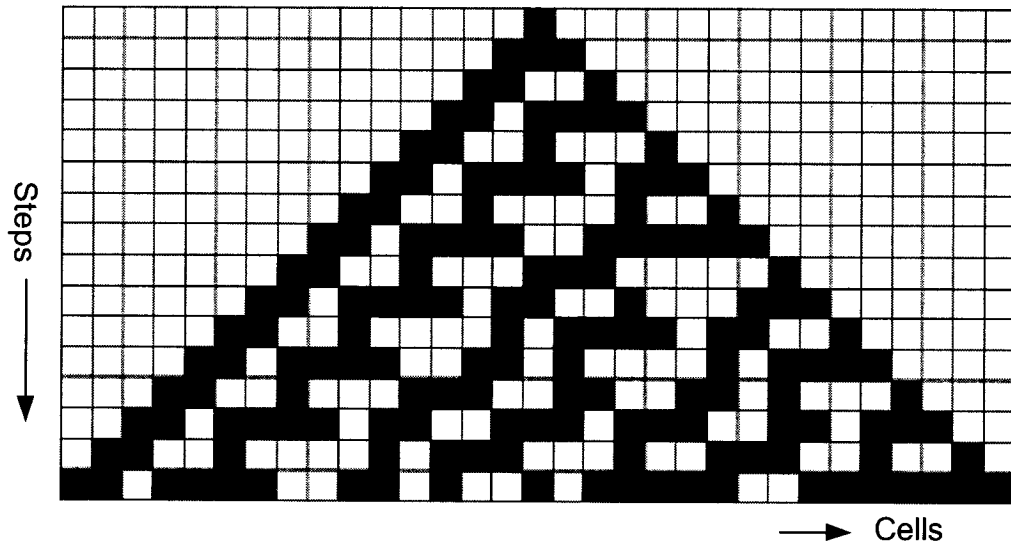


Figure 2.2: The evolved pattern after 15 steps [26]

More complex patterns can emerge by using cellular automata of higher orders.

The main reason that attracts researchers to cellular automata is its impressive feature of emergence. It is a property of complex systems that has recently received much attention and has played an essential role in developing “intelligence” without complications. Although there is no formal definition of the emergence, it is highly associated with the scale as referred by P. Anderson in his interesting article “*More is different.*” [27] He stated that “the behavior of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple exploration of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other.” [27] For example, on a molecular scale,

a single molecule of  $H_2O$  does not tell anything about liquidity, but a collection of millions of molecules forms water [28]. On neuron scale, the study of one neuron does not suggest consciousness but a collocation of millions of interacting neurons creates consciousness [28]. Swarms and animal aggregations are also eligible for that; behavior of solo individuals as a part, tells nothing about the stunning achievement that can be done by the whole. Attaining goals via this approach is attributed to the intelligence that emerges by interactions within a group of primitive individuals and is known as swarm intelligence. Particle swarm optimizer (PSO) falls under this category.

Like cellular automata, in the PSO, intentions are not explicitly coded. Instead, solutions emerge by updating the positions of individuals according to their own experience and attitudes acquired from others. Unlike cellular automata, the state of individuals in the PSO can take continuous values. Also, the locality is not an exclusive property of the neighborhood in the PSO since states of individuals outside adjacent neighborhoods can be also considered.

### 2.3 Original particle swarm optimizer

A particle swarm optimizer is a population-based search technique that aims at solving nonlinear multi-model optimization problems without the need to calculate derivatives of objective functions. This technique was inspired by simulated flocking models in artificial life science. Kennedy and Eberhart [3] noticed the analogy between birds flocking and function optimization, especially at times when birds cluster around sites that they have no prior knowledge about. To simulate their idea for function optimization, they assumed a simulated site at the coordinate (100,100) in a two dimensional XY grid and set their swarm to find this site. The nearness (fitness) of an individual to the simulated target is calculated using a so-called “cornfield vector” (Equation (2-2)).

$$f_k^i = \sqrt{(X_k^i - 100)^2} + \sqrt{(Y_k^i - 100)^2} \quad (2-2)$$

Where  $(X_k^i, Y_k^i)$ : the position coordinates that correspond to the fitness value  $f_k^i$  for the  $i^{th}$  individual at  $k^{th}$  iteration.

Individuals also remember their best fitness value  $F^i$  (best position reached by this individual) and its coordinates  $(X^i, Y^i)$ . Individuals compare their current position  $(X_k^i, Y_k^i)$  with their best position  $(X^i, Y^i)$ . In  $X$  direction, if the current position is situated to the right of the best position; the position in the next step is updated by reducing the speed in  $X$  direction ( $VX$ ) by the amount  $c_1 rand$  according to Equation (2-3) ( $c_1$  is a system parameter taking constant value and  $rand$  is a uniformly random variable between 0 and 1). If the current position is located to the left of the best position; the position in the next step is updated by increasing the speed in  $X$  direction ( $VX$ ) by the amount  $c_1 rand$  according to Equation (2-4).

$$\text{if } X_k^i > X^i \quad \therefore \quad VX_{k+1}^i = VX_k^i - c_1 rand \quad (2-3)$$

$$\text{if } X_k^i < X^i \quad \therefore \quad VX_{k+1}^i = VX_k^i + c_1 rand \quad (2-4)$$

The position in  $Y$  direction is also updated up and down by adjusting the speed in  $Y$  direction ( $VY$ ) as described by Equations (2-5) and (2-6).

$$\text{if } Y_k^i > Y^i \quad \therefore \quad VY_{k+1}^i = VY_k^i - c_1 rand \quad (2-5)$$

$$\text{if } Y_k^i < Y^i \quad \therefore \quad VY_{k+1}^i = VY_k^i + c_1 rand \quad (2-6)$$

Moreover, individuals have access to the global best fitness  $F_k^g$  that has been found by one of individuals and the corresponding position  $(X_k^g, Y_k^g)$ . They compare their current position  $(X_k^i, Y_k^i)$  with the global best position  $(X_k^g, Y_k^g)$ . Then, the position is updated by the same manner using Equations (2-7) to (2-10) as described above. The value of the system parameters  $c_1$  and  $c_2$  was set to 2.

$$\text{if } X_k^i > X_k^g \quad \therefore \quad VX_{k+1}^i = VX_k^i - c_2 rand \quad (2-7)$$

$$\text{if } X_k^i < X_k^g \quad \therefore \quad VX_{k+1}^i = VX_k^i + c_2 rand \quad (2-8)$$

$$\text{if } Y_k^i > Y_k^g \quad \therefore \quad VY_{k+1}^i = VY_k^i - c_2 rand \quad (2-9)$$

$$\text{if } Y_k^i < Y_k^g \quad \therefore \quad VY_{k+1}^i = VY_k^i + c_2 rand \quad (2-10)$$

In their simulation, Kennedy and Eberhart used swarm sizes between 15 to 30 individuals. Surprisingly the flock was able to cluster around the simulated cornfield that was marked by a circle in the  $XY$  pixel plane after a few iterations. Instead of the crude inequality test



used to adjust positions, the authors suggested a simplified version based on the differences between current positions as well as both personal best positions and the global best position as described by Equations (2-11) to (2-14) where  $c_1rand$  and  $c_2rand$  are dimensionless. The iterative process was terminated when successive values of global best solution were fixed or the difference between them was less than or equal to a pre-specified tolerance  $\varepsilon$  (Equation (2-15)).

$$VX_{k+1}^i = VX_k^i + c_1rand \frac{(X^i - X_k^i)}{\Delta t} + c_2rand \frac{(X_k^g - X_k^i)}{\Delta t} \quad (2-11)$$

$$VY_{k+1}^i = VY_k^i + c_1rand \frac{(Y^i - Y_k^i)}{\Delta t} + c_2rand \frac{(Y_k^g - Y_k^i)}{\Delta t} \quad (2-12)$$

$$X_{k+1}^i = X_k^i + VX_{k+1}^i \Delta t \quad (2-13)$$

$$Y_{k+1}^i = Y_k^i + VY_{k+1}^i \Delta t \quad (2-14)$$

$$|F_k^g - F_{k-q}^g| \leq \varepsilon \quad q = 1, 2, \dots, S \quad (2-15)$$

Due to its simplicity, PSO has attracted many researchers from different disciplines. In an online search on the IEEE electronic library alone, from the year 1995 to the end of 2006, the query “swarm optimizer” has produced over a 1000 articles. For the first 7 years (1995-2002) the appetite to explore the technique was not that big. From 2003 to 2006, the number of published works in the PSO has dramatically increased, as shown in Figure 2.3.

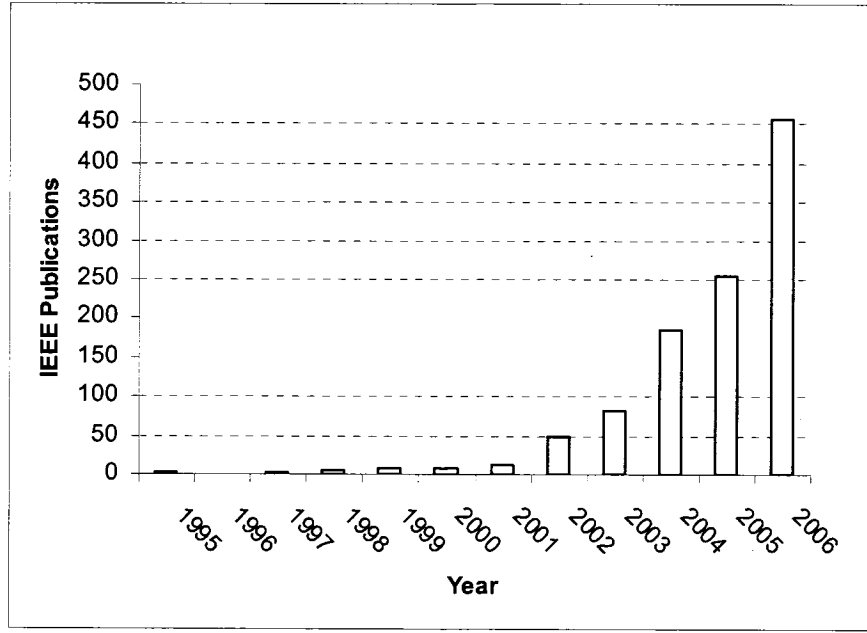


Figure 2.3: PSO publications from 1995-2006 (IEEE).

The PSO has been applied to solve optimization problems in several research areas. In many cases, the application of the PSO is accompanied with some modifications. These modifications always target the position update equation (Equations (2-11) and (2-12)). In the next subsections, some PSO variants, modifications, and issues related to PSO are reviewed.

## 2.4 Inertia weight

Inertia weight factor is the first significant modification that has been made to the PSO. The factor was proposed by Shi and Eberhart [29] to play a balancing role between local search and global search abilities. According to the authors, the inertia weight can be a constant value, linear, or nonlinear time-variant functions. The inertia weight ( $W$ ) is introduced to the position update equation as described in Equation (2-16).

$$VX_{k+1}^i = W * VX_k^i + c_1 rand \frac{(X^i - X_k^i)}{\Delta t} + c_2 rand \frac{(X_k^g - X_k^i)}{\Delta t} \quad (2-16)$$

Simulation results that have been obtained by the authors suggested that inertia weight values in the range of [0.9, 1.2] would lead to a better performance. A linear time-decreasing inertia was also tested. A significant improvement of the performance was claimed by authors in the case of the decreasing inertia factor [29].

In reference [30], the same authors compared the impact of a constant inertia and a linearly decreasing inertia on the performance of the PSO. They found that the decreasing inertia weight starting at a value of 1.4 at the beginning of the search and diminishing to zero by the end, is better than the fixed value inertia. According to the authors, a higher value of inertia at the beginning of the search facilitates wider exploration of the domain while the small value at the end allows fine tuning. They finally concluded that the proper value of the time decreasing inertia would start at 0.9 and end at 0.4.

In contrast, Zhang et al. [31,32] have empirically tested a linearly time-increasing inertia weight for the same purpose. Their setting for inertia weight was the opposite of the setting proposed in reference [30]; the inertia weight is set to start at 0.4 and end at 0.9. They argued that both global and local searches can benefit from the small inertia weight. In that sense, assuming a small inertia weight at the beginning will not do any harm to the global search required at that stage. Moreover, a large inertia weight can help stabilize the algorithm at the end of the search process when it is needed most. The authors were able to get better results than those obtained using the decreasing inertia weight in terms of accuracy and speed of convergence. However, decreasing the inertia weight seems appealing to the majority of people who work with the PSO.

It seems that neither of the two strategies described above would be wrong. Nevertheless, the apparent contradiction is just due to lack of hypothetical inference and an expected implication of the No Free Lunch theory (NFL:“....for any algorithm, any elevated performance over one class of problems is offset by worse performance over another class” [33]).

Nonlinear classes of inertia weights have been also explored. In reference [34] the authors proposed a nonlinearly decreasing inertia weight in the form described in Equation (2-17).

$$W_k = \left[ \frac{k_{\max} - k}{k_{\max}} \right]^n * (W_{\text{int}} - W_{\text{fin}}) + W_{\text{fin}} \quad (2-17)$$

Where:

$W_k$ : inertia weight at iteration  $k$

$k$ : iteration index

$n$ : a user defined modulation index

$k_{\max}$ : maximum allowable iteration number

$W_{\text{int}}$  and  $W_{\text{fin}}$ : initial and final values of inertia weight factor respectively.

The authors also defined a so-called inertia slope  $m = (W_{\text{fin}} - W_{\text{int}}) / k_{\max}$ . Here, the PSO system can be identified by a set of three parameters  $\{W_{\text{int}}, m, n\}$ ; giving that all other parameters are kept untouched. For  $n=1$ , the inertia weight factor will be a linearly decreasing function as the one described in [29] and [30]. The authors reported that a successful tuning of the three specified parameters could lead to better solution. As a result of their simulation, they suggested that values of  $W_{\text{int}} = 0.2$ ,  $m = -2.5 \times 10^{-4}$ , and  $n = 1.2$  would be satisfactory.

Lei et al. [35] proposed a PSO version with an inertia weight factor that can take multiple values within the same population. Different values for inertia are generated by using the Sugeno complement operator as described in Equation (2-18).

$$W_k = \frac{1 - (k / k_{\max})}{1 + (s * k / k_{\max})}; \quad s > -1 \quad (2-18)$$

In that case, a set of monotonic inertia weight curves can be generated by assuming different values for the parameter  $s$  in Equation (2-18). For example, a value of  $s=0$  would result in a linearly decreasing inertia weight.

It is not clear from reference [35], on what basis individuals are assigned to different inertia weights. Apparently, the entire swarm is divided into equal sub-swarms and each one is run with a different inertia weight. According to the authors, over the course of iteration, global optima obtained by sub-swarms are examined at every definite interval. Then, the inertia weight that results in the worst global optimum is replaced by the one that produces the best global optimum. This process is repeated until the suitable inertia

weight dominates the entire swarm. The authors claimed that this strategy was able to speed up the convergence of the PSO.

Natural exponential forms were also among the functions that have been reported in the literature for the inertia weights. In two independent papers, Chen et al. [36,37] proposed several natural exponential functions for the inertia weight factor. Equation (2-19) describes a so-called self-active inertia where inertia is adjusted adaptively according to the fitness progress rate  $\Delta(k)$ .

$$W_k = W_{fin} + (W_{int} - W_{fin}) * e^{\frac{\Delta(k)}{\Delta(k-1)}} \quad (2-19)$$

$$\Delta(k) = |f(k) - f(k-1)| \quad (2-20)$$

The other two functions are described by Equations (3-21) and (3-22). Initial and final values of inertia weight are often assumed to be 0.9 and 0.4 respectively as recommended in reference [30].

$$W_k = W_{fin} + (W_{int} - W_{fin}) * e^{-k / (\frac{k_{max}}{10})} \quad (2-21)$$

$$W_k = W_{fin} + (W_{int} - W_{fin}) * e^{-[k / \frac{k_{max}}{4}]^2} \quad (2-22)$$

## 2.5 Analysis of the particle swarm

Over time, a misleading fact about the PSO being a global optimizer was established. Yet, there is no analytical proof that the simple paradigm of the PSO can even converge to a local optimum. Moreover, the stability of the PSO is not guaranteed unless special damping factors are incorporated. Due to its stochastic nature, the so-called “drunkard walks” [38] of the particles during the course of the search process might cause the PSO system to “explode.” Explosion can be in the form of exploring areas outside the domain. This can also take place when the system does not terminate in a finite number of steps. It is known that limiting the steps taken by particles to the upper bounds of the problem’s variables is a good choice to avoid instability.

### 2.5.1 Trajectory analysis

Ozcan and Mohan [39,40] have introduced the first study on the behavior of PSO by analyzing the trajectory of motion. Although they did not specifically comment on the convergence of the PSO, their work gave some insight into possible trajectories of particles during the search process. The authors analyzed the behavior of a simple PSO (no inertia factor) described by Equations (2-23) and (2-24). For simplicity, dimensions in velocity and position update equations at a given time  $t$  for an individual  $i$  are reduced to one dimension.

$$V_i(t) = V_i(t-1) + \varphi_1[l_i(t) - X_i(t-1)] + \varphi_2[g(t) - X_i(t-1)] \quad (2-23)$$

$$X_i(t) = X_i(t-1) + V_i(t) \quad (2-24)$$

where

$g(t)$  and  $l(t)$  are the global best and personal best positions respectively.

$$\varphi_1 = c_1 * rand$$

$$\varphi_2 = c_2 * rand$$

Further assumptions about the PSO system have been made by the authors to make the analysis more tractable.

*Assumptions:*

- 1- Deterministic PSO: The stochastic nature of the PSO was neutralized by assuming constant values for  $\varphi_1$  and  $\varphi_2$ ;  $\varphi_1 = \varphi_2 = constant$ .
- 2- No interaction: Analysis was made for a single particle (symbol  $i$  is dropped).
- 3- Global best and personal best positions were kept constant and equal during the analysis;  $l = g = constant = p$

Accordingly, Equations (2-23) and (2-24) can be simplified further as in Equations (2-26) and (2-27).

$$V(t) = V(t-1) + \varphi_1[p - X(t-1)] + \varphi_2[p - X(t-1)] \quad (2-25)$$

$$V(t) = V(t-1) - \varphi X(t-1) + \varphi p \quad (2-26)$$

$$X(t) = X(t-1) + V(t) \quad (2-27)$$

where  $\varphi_1 + \varphi_2 = \varphi$

By substitution between Equations (2-26) and (2-27), a non-homogenous linear recurrence Equation (2-28) is obtained.

$$X(t) - (2 - \varphi)X(t-1) + X(t-2) = \varphi p \quad (2-28)$$

A solution to the position update equation expressed by recurrence Equation (2-28) is described by the closed form (2-29), assuming initial conditions of  $V(0) = v_0$ ;  $X(0) = x_0$  and  $X(1) = x_0(1 - \varphi) + v_0 + \varphi p$ .

$$X(t) = \alpha Z_1^t + \beta Z_2^t + p \quad (2-29)$$

where;

$$Z_1 = \frac{2 - \varphi + \delta}{2}$$

$$Z_2 = \frac{2 - \varphi - \delta}{2}$$

$$\delta = \sqrt{\varphi^2 - 4\varphi}$$

$$\beta = (x_0 - p)(\delta + \varphi)/(2\delta) - v_0/\delta$$

$$\alpha = x_0 - p - \beta$$

The shape of the particle's trajectory is completely defined by the value of  $\delta$  providing that other system parameters are kept constant.

Real-valued  $\delta$  ( $\varphi > 4$ ):

Assuming that the particle starts at  $X(0) = x_0 = p = 0$ ; the trajectory Equation (2-29) would be as described by Equation (2-30).

$$X(t) = \left(\frac{v_0}{\delta}\right) \left[ \left(\frac{2 - \varphi + \delta}{2}\right)^t - \left(\frac{2 - \varphi - \delta}{2}\right)^t \right] \quad (2-30)$$

It is clear from Equation (2-30) that the trajectory is described by an increasing exponential function that might cause the particle to move outside the domain.

Complex-valued  $\delta$  ( $\varphi < 4$ )

Imposing the same initial condition as above; values of  $Z_1$ ,  $Z_2$ ,  $\delta$ ,  $\beta$ , and  $\alpha$  will be complex and the trajectory of the motion in this case can be described as in Equation (2-31).

$$Z_1 = \frac{2 - \varphi + \delta'}{2}$$

$$Z_2 = \frac{2 - \varphi - \delta'}{2}$$

$$\delta' = i\sqrt{|\varphi^2 - 4\varphi|}$$

$$\beta = -v_0 / \delta'$$

$$\alpha = -\beta$$

$$X(t) = (2v_0 / \|\delta'\|) \sin \theta \quad (2-31)$$

$$\text{where } \theta = \tan^{-1} \left( \frac{\|\delta'\|}{2 - \varphi} \right)$$

In this case, the trajectory is a sinusoidal wave. As a result, the particle could periodically visit the same regions that have been already searched before.

At the root ( $\varphi = 0$ )

The recursive Equation (2-28) will be:

$$X(t) - 2X(t-1) + X(t-2) = 0 \quad (2-32)$$

The solution yields the closed form 2-33.

$$X(t) = (x_0 - p) + v_0 t \quad (2-33)$$

Assuming  $X(0) = x_0 = p = 0$ , the particle in this case will keep moving with the initial velocity until the end.

At the root ( $\varphi = 4$ )

In this case, the recursive Equation (2-28) can be written as in Equation (2-34) with a closed trajectory form (2-35).

$$X(t) + 2X(t-1) + X(t-2) = 4p \quad (2-34)$$

$$X(t) = ((x_0 - p) + (2(x_0 - p) - v_0)t)(-1)^t + p \quad (2-35)$$

Imposing  $X(0) = x_0 = p = 0$ ; the trajectory will yield the form  $X(t) = -v_0 t (-1)^t$ .

In this case the particle will move back and forth with the initial velocity.

### 2.5.2 Stability analysis

Kennedy and Clerc [38] have also studied the behavior of the PSO. Their study was aimed at examining whether the simple PSO is able to eventually terminate at an



arbitrary point or not. Alternatively, they examined the velocity of particles, which supposedly converges to zero after a definite time at that point. The convergence point is not necessarily a local or a global optimum. Again, the authors assumed a one-dimensional single-particle system for their analysis. They deduced a slightly different representation that will be reproduced in this section. The model is a state space one, which is completely defined by the eigenvalues of its state matrix. Assuming constant  $l$ ,  $g$ ,  $\varphi_1$  and  $\varphi_2$ , the velocity of a single particle in one dimension at time  $t+1$  is repeated below:

$$V(t+1) = V(t) + \varphi_1[l - X(t)] + \varphi_2[g - X(t)]$$

$$V(t+1) = V(t) + \varphi_1 l - \varphi_1 X(t) + \varphi_2 g - \varphi_2 X(t)$$

$$V(t+1) = V(t) + (\varphi_1 + \varphi_2)[(\varphi_1 l + \varphi_2 g)/(\varphi_1 + \varphi_2) - X(t)]$$

The authors made the assumption that  $p = (\varphi_1 l + \varphi_2 g)/(\varphi_1 + \varphi_2)$  and  $\varphi = \varphi_1 + \varphi_2$ .

Accordingly, the velocity update equation will be:

$$V(t+1) = V(t) + \varphi[p - X(t)]$$

With  $y(t) = p - X(t)$ , the velocity update equation is now:

$$V(t+1) = V(t) + \varphi y(t) \quad (2-36)$$

The same arrangements are made to the position update equation; as described below:

$$X(t+1) = X(t) + V(t+1)$$

Substituting by the value of  $V(t+1)$  from Equation (2-36);

$$X(t+1) = X(t) + V(t) + \varphi y(t)$$

Multiplying by  $-1$  and adding  $p$  to both sides of the equation;

$$p - X(t+1) = p - X(t) - V(t) - \varphi y(t)$$

$$\therefore y(t+1) = y(t) - V(t) - \varphi y(t)$$

$$y(t+1) = -V(t) + (1 - \varphi)y(t) \quad (2-37)$$

This results in a simple dynamic system specified by Equations (2-36) and (2-37) with a state matrix  $M$  as shown below.

$$V(t+1) = V(t) + \varphi y(t)$$

$$y(t+1) = -V(t) + (1 - \varphi)y(t)$$

$$M = \begin{bmatrix} 1 & \varphi \\ -1 & 1 - \varphi \end{bmatrix}$$

The authors studied this model and they found that, in order for this system to converge the condition of  $\max(|e_1|, |e_2|) < 1$ ; where  $e_1$  and  $e_2$  are the eigenvalues of  $M$ , has to be satisfied. Again, the concept of convergence that has been reported to this point is just the convergence on its weakest form or the stability of the PSO as a dynamic system. A dynamic system is said to be convergent if  $\lim_{t \rightarrow \infty} V(t) = 0$  at an arbitrary point  $A$ ; where  $\lim_{t \rightarrow \infty} X(t) = A$ . Accordingly, another model with a constricted velocity to prevent divergence was proposed. The idea was to introduce a PSO system with eigenvalues  $e_1'$  and  $e_2'$  that already satisfy  $\max(|e_1'|, |e_2'|) < 1$ .

The constricted velocity update equation was described as:

$$V(t+1) = \chi[V(t) + \varphi y(t)]$$

where  $\chi$  is the constriction factor.

Therefore, the position update equation has become

$$X(t+1) = X(t) + \chi[V(t) + \varphi y(t)]$$

$$p - X(t+1) = p - X(t) - \chi[V(t) + \varphi y(t)]$$

$$y(t+1) = y(t) - \chi V(t) - \chi \varphi y(t)$$

$$y(t+1) = -\chi V(t) + (1 - \chi \varphi)y(t)$$

The new constrained state space model is now described by Equations (2-38) and (2-39) and  $M'$ .

$$V(t+1) = \chi V(t) + \chi \varphi y(t) \quad (2-38)$$

$$y(t+1) = -\chi V(t) + (1 - \chi \varphi)y(t) \quad (2-39)$$

$$M' = \begin{bmatrix} \chi & \chi \varphi \\ -\chi & 1 - \chi \varphi \end{bmatrix}$$

To gain more control over the convergence rate they defined 5 parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ , and  $\eta$  for the system as below:

$$V(t+1) = \alpha V(t) + \beta \varphi y(t)$$

$$y(t+1) = -\gamma V(t) + (\delta - \eta \varphi)y(t)$$

Kennedy and Clerc derived several formulae for the constriction factor based on different settings for the five control parameters. The advantage of using the constriction factor is that the need to confine the steps taken by particles by the maximum limits of the domain variables is now eliminated. Although all proposed constriction factors have the same purpose, which is avoiding instability by constricting the velocity from growing excessively, they vary in their effect on the convergence rate. The authors suggested employing a moderate constriction that allows exploration of the entire domain and prevents premature convergence to an unwanted point. The most common constriction factor is described as:

$$\chi = \frac{2k}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}, \quad \text{with } \varphi = c_1 + c_2 \geq 4 \text{ and } k \in [0, 1]$$

The trajectory of a simple PSO with inertia weight was also analyzed by F. Bergh [41] to provide guidelines for the choice of the parameters  $c_1$ ,  $c_2$ , and  $W$ . In addition to insights about the stability of the PSO, he also addressed the ability of the system to converge to local and global optima.

The velocity update equation for the one-dimensional single particle at time  $t+1$  was also considered. So, the velocity of that particle can be specified by:

$$V(t+1) = WV(t) + c_1 r_1(t)[l(t) - X(t)] + c_2 r_2(t)[g(t) - X(t)]$$

where  $r_1$  and  $r_2$  are uniformly distributed random numbers between 0 and 1.

Ignoring randomness and assuming constant  $l$ ,  $g$  and  $W$ , the equation can be rewritten as:

$$V(t+1) = WV(t) + \varphi_1[l - X(t)] + \varphi_2[g - X(t)]$$

Hence, the position update equation becomes:

$$X(t+1) = X(t) + WV(t) + \varphi_1[l - X(t)] + \varphi_2[g - X(t)]$$

$$\therefore X(t+1) = (1 - \varphi_1 - \varphi_2)X(t) + WV(t) + \varphi_1 l + \varphi_2 g$$

Substituting by  $V(t) = X(t) - X(t-1)$

$$\therefore X(t+1) = (1 + W - \varphi_1 - \varphi_2)X(t) + WX(t-1) + \varphi_1 l + \varphi_2 g$$

The solution to this non-homogenous recurrence relation, assuming initial conditions  $X(0) = x_0$  and  $X(1) = x_1$  was given as:

$$X(t) = k_1 + k_2 \alpha^t + k_3 \beta^t \quad (2-40)$$

where:

$$k_1 = \frac{\varphi_1 l + \varphi_2 g}{\varphi_1 + \varphi_2} \quad (2-41)$$

$$\gamma = \sqrt{(1 + W - \varphi_1 - \varphi_2)^2 - 4W} \quad (2-42)$$

$$\alpha = \frac{1 + W - \varphi_1 - \varphi_2 + \gamma}{2} \quad (2-43)$$

$$\beta = \frac{1 + W - \varphi_1 - \varphi_2 - \gamma}{2} \quad (2-44)$$

$$x_2 = (1 + W - \varphi_1 - \varphi_2)x_1 + Wx_0 + \varphi_1 l + \varphi_2 g \quad (2-45)$$

$$k_2 = \frac{\beta(x_0 - x_1) - x_1 + x_2}{\gamma(\alpha - 1)} \quad (2-46)$$

$$k_3 = \frac{\alpha(x_1 - x_0) + x_1 - x_2}{\gamma(\beta - 1)} \quad (2-47)$$

From Equation (2-40), in order to have convergence, the condition  $\max(\|\alpha\|, \|\beta\|) < 1$  has to be met. If the convergence is made available, the PSO system will converge to the point  $X = k_1 = \frac{\varphi_1 l + \varphi_2 g}{\varphi_1 + \varphi_2}$ ; where  $\lim_{t \rightarrow \infty} X(t) = k_1 + k_2 \alpha' + k_3 \beta' = k_1$ .

Assuming convergence at

$$X = k_1 = \frac{\varphi_1 l + \varphi_2 g}{\varphi_1 + \varphi_2}$$

$$\therefore X = \frac{\varphi_1}{\varphi_1 + \varphi_2} l + \frac{\varphi_2}{\varphi_1 + \varphi_2} g$$

$$X = (1 - \frac{\varphi_2}{\varphi_1 + \varphi_2}) l + \frac{\varphi_2}{\varphi_1 + \varphi_2} g$$

$$X = (1 - a)l + ag$$

$$\text{where } a = \frac{\varphi_2}{\varphi_1 + \varphi_2} \in [0, 1]$$

This means that the system will converge to a point  $X$ , which is a weighted mean of personal best  $l$  and global best  $g$ .

The criterion for selecting  $c_1$ ,  $c_2$ , and  $W$  that ensures the convergence of the PSO was empirically proven to be:

$$W > \frac{1}{2}(c_1 + c_2) - 1 \quad (2-48)$$

Example for non-convergent parameters setting (original PSO)

The author tested the convergence of the originally developed PSO ( $c_1=2$ ,  $c_2=2$  and  $W=1$ ) using his findings. It is clear that the criterion in Equation (2-48) was violated since  $\frac{1}{2}(c_1 + c_2) - 1 = 1 = W$ .

Also the condition of whether  $\max(\|\alpha\|, \|\beta\|) < 1$  or not was also examined by calculating values of  $\alpha$  and  $\beta$  as follows

Substituting  $\varphi = \varphi_1 + \varphi_2$  and  $W=1$  into Equation (2-42).

$$\gamma = \sqrt{(2 - \varphi)^2 - 4}$$

$$\therefore \gamma = i\sqrt{4\varphi - \varphi^2}$$

Since  $\gamma$  is a complex,  $\alpha$  and  $\beta$  become complex conjugates and are calculated as in Equations (2-43) and (2-44).

$$\alpha = \frac{2 - \varphi + i\sqrt{4\varphi - \varphi^2}}{2}$$

$$\alpha = \frac{2 - \varphi}{2} + i\frac{\sqrt{4\varphi - \varphi^2}}{2}$$

$$\therefore \|\alpha\| = \sqrt{\frac{(2 - \varphi)^2}{4} + \frac{4\varphi - \varphi^2}{4}} = 1$$

This means that  $\|\beta\| = \|\alpha\| = 1$ , therefore the condition for convergence ( $\max(\|\alpha\|, \|\beta\|) < 1$ ) is violated and the system will diverge regardless of the value of  $\varphi$ . This explains the importance of limiting the velocity in this setting to  $V_{max}$  in order to reach a convergence state.

Due to the fact that  $\alpha$  and  $\beta$  are complex, the PSO trajectory in this case yields the sinusoidal wave described below.

$$X(t) = k_1 + k_2[\cos\theta + i\sin\theta] + k_3[\cos\theta - i\sin\theta]$$

where  $\theta = \arg(\alpha) = -\arg(\beta)$

Although the trajectory equation seems bounded, the convergence takes place due to the oscillation effect that has been resulted from the sinusoidal wave. Thus, the particle is not flying in hyperspace but rather searching the problem domain by surfing sinusoidal waves [40].

#### Example for convergent parameters setting

F. Bergh [41] gave the following example for convergent PSO parameters; a choice of  $c_1=c_2=1.49618$  and  $W=0.7298$ . It is clear that the relation 2-48 is satisfied in this case since  $\frac{1}{2}(c_1 + c_2) - 1 = 0.49618 < W$ . To examine the condition for convergence

( $\max(\|\alpha\|, \|\beta\|) < 1$ );  $\gamma$ ,  $\alpha$  and  $\beta$  are calculated

$$\gamma = \sqrt{(1 + W - \varphi)^2 - 4W}$$

$$\gamma = \sqrt{(1 + 0.7298 - \varphi)^2 - 4(0.7298)}$$

$$\gamma = \sqrt{(\varphi - 0.02122)(\varphi - 3.4384)}$$

There are two cases:

Case 1:  $\gamma$  is real if

( $\varphi \geq 0.02122$ ) and ( $\varphi \geq 3.4384$ ), which is not acceptable because  $\varphi \in (0, c_1 + c_2)$  when  $r_1, r_2 \in U[0,1]$ .

Or

( $\varphi \leq 0.02122$ ) and ( $\varphi \leq 3.4384$ )

So  $\gamma$  is real for values of  $\varphi \in [0, 0.02122]$ .

Accordingly, for real-valued  $\gamma$

$$\max(\|\alpha\|, \|\beta\|) = \alpha = \frac{1.7298 - \varphi + \sqrt{(\varphi - 0.02122)(\varphi - 3.484)}}{2} < 1$$

Case 2:  $\gamma$  is complex if

( $\varphi < 0.02122$ ) and ( $\varphi > 3.4384$ ), which is not acceptable.

Or

( $\varphi > 0.02122$ ) and ( $\varphi < 3.4384$ )

So  $\gamma$  is complex for values of  $\varphi \in (0.02122, 2.992]$  (note that, for this setting the value of  $\varphi$  is limited to  $c_1+c_2=2.992$  rather than 3.4384).

Accordingly, for complex-valued  $\gamma$

$$\alpha = \frac{1+W-\varphi+i\sqrt{4W-(1+W-\varphi)^2}}{2}$$

$$\beta = \frac{1+W-\varphi-i\sqrt{4W-(1+W-\varphi)^2}}{2}$$

$$\max(\|\alpha\|, \|\beta\|) = \|\alpha\| = \|\beta\| = \sqrt{\frac{(1+W-\varphi)^2}{4} + \frac{4W-(1+W-\varphi)^2}{4}} = 0.8542 < 1$$

Therefore, a PSO system with this setting will converge for all values of  $\varphi$  providing that  $c_1=c_2=1.49618$  and  $W=0.7298$  without the need to limit the velocity to  $V_{max}$ .

### 2.5.3 Stability of the stochastic PSO

From the above discussion, it can be concluded that the generic PSO does not converge unless certain precautions such as velocity clamping and careful choosing of parameters are made to ensure the stability. Although those kinds of behavior analysis have provided valuable information about the technique, none of them reflects the reality of the PSO. Assuming deterministic PSO by ignoring the stochastic behavior demonstrated by  $r_1$  and  $r_2$ , has caused a loss of precision in the studied models. Moreover, assuming time-invariant values for  $l$  and  $g$  does not seem to be the case with the PSO. Practically, these variables are allowed to change over the course of iterations.

It was not until recently that the stochastic nature of the PSO has been taken into consideration with the issue of convergence. In reference [42], the PSO was analyzed using the stochastic process theory. The particle's position is considered as a stochastic vector. First the authors adopted the one-dimensional single particle model and then the analysis was relaxed to include D-dimensional, M-particle representation. Also, the authors allowed both the global best position ( $g$ ) and the personal best position ( $l$ ) to change over the time as  $\vec{g}(t)$  and  $\vec{l}(t)$  respectively. The convergence of expectation and variance of the random variable representing the particle's position was employed to

address the convergence of the system itself. In this case the particle swarm system is said to be convergent if:

$\bar{X}_i(t)$  converges in mean square to  $\bar{P} \quad \forall i \in \{1, 2, \dots, M\}$

Or yet,  $\lim_{t \rightarrow \infty} E|\bar{X}_i(t) - \bar{P}|^2 = 0$ .

Where:

$\bar{X}_i(t)$  is the position of  $i^{th}$  particle at time  $t$

$E\bar{X}_i(t)$  is the expectation of  $\bar{X}_i(t)$

$\bar{P}$  is an arbitrary position in the search space

$M$  is size of the population

It also implies that in order for  $\bar{X}_i(t)$  to converge in mean square to  $\bar{P}$ ,  $\lim_{t \rightarrow \infty} E\bar{X}_i(t) = \bar{P}$  and

$\lim_{t \rightarrow \infty} E\bar{D}_i(t) = 0$  have to be simultaneously satisfied, where  $E\bar{D}_i(t)$  is the variance.

The study found that the particle swarm system will converge in mean square to  $\bar{g}$  when the following limitations are satisfied:

$$W, c_1, c_2 \geq 0$$

$$0 \leq W < 1$$

$$c_1 + c_2 > 0 \text{ and}$$

$$0 < f(1) < \frac{c_2^2(1+W)}{6} \text{ where}$$

$$f(1) = -(c_1 + c_2)W^2 + \left(\frac{1}{6}c_1^2 + \frac{1}{6}c_2^2 + \frac{1}{2}c_1c_2\right)W + c_1 + c_2 - \frac{1}{3}c_1^2 - \frac{1}{3}c_2^2 - \frac{1}{2}c_1c_2$$

Although the resulting relations seem very restrictive, settings that are empirically proven to be good choices fall within these limits [42].

A stability analysis of the stochastic particle dynamics was also presented in [43]. A less generalized model was considered by assuming the position equation for the best particle (*i.e.*  $l=g$ ). During the analysis, the value of  $l$  was assumed to be time invariant equal to a constant  $p$  ( $l=g=p$ ). Sufficient conditions for stability were driven by using the Lyapunov stability analysis. The stochastic PSO system was described by a set of Equations (2-49) – (2-51).



$$\xi(t+1) = A\xi(t) + Bu(t) \quad (2-49)$$

$$y(t) = C\xi(t) \quad (2-50)$$

$$u(t) = -\varphi(t)y(t) \quad (2-51)$$

where  $\xi(t) = \begin{pmatrix} X(t) - p \\ V(t) \end{pmatrix}$ ,  $A = \begin{pmatrix} 1 & W \\ 0 & W \end{pmatrix}$ ,  $B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ ,  $C = \begin{pmatrix} 1 & 0 \end{pmatrix}$ , and

$$\varphi(t) = \varphi_1(t) + \varphi_2(t), \quad 0 < \varphi(t) < c_1 + c_2$$

the Lyapunov stability theory is based on the fact that the total energy function of the system should be monotonically decreasing in order for this system to be asymptotically stable (i.e.  $f(\xi(t+1)) - f(\xi(t)) < 0$ , where  $f$  is a Lyapunov function). The authors proved that the system described by (2-49) – (2-51) has a unique equilibrium point at  $p$ , which is asymptotically stable if:

$$|W| < 1, \quad W \neq 0 \text{ and } c_1 + c_2 < \left( \frac{2(1 - 2|W| + W^2)}{1 + W} \right)$$

This analysis is valid only for the best particle whose personal best position is the same as the global best position and is not applicable for any other particle. The authors stated that “violation of the stability conditions does not imply instability; rather that stability cannot be guaranteed.”

#### 2.5.4 Local and global convergence

In the earlier sections, analytical studies concerning the stability of the PSO and convergence to an arbitrary point that does not necessarily coincide with local or global optima are reviewed. In this section, a different prospective of convergence is reviewed.

The PSO system converges to a local/global optimum if:

$$\lim_{k \rightarrow \infty} \{g(k)\} = X^*, \quad X^* \in R_{opt}$$

Where:

$\{g(k)\}$  is the sequence of the generated global best positions

$R_{opt}$  is the optimality region (locally or globally)

The ability of the PSO algorithm to converge to local and global optima was discussed in [41], [44] and [45]. The authors of these references used the criteria derived by Solis and

Wets [46] for stochastic search techniques to study the convergence of the PSO. For convenience, Solis and Wets theorems on convergence are informally reproduced here from reference [41]. Solis and Wets proposed three conditions for the convergence of stochastic search algorithms. Specific combinations of these conditions will determine the type of convergence.

- $H_1$ : a new solution constructed by the algorithm should be no worse than the current solution.
- $H_2$ : the probability of repeatedly missing any sub-area  $s \in S$  (where  $S$ : the area of the entire search space) during the search should be zero, which implicitly means that the probability of visiting areas inside optimality region  $R_{opt}$  should be nonzero.
- $H_3$ : during the search process, the algorithm is able at every step to move an arbitrary point  $z$  closer to optimality region by at least distance  $\gamma$  with probability greater than or equal to nonzero  $\eta$ .

Thus, the theorems of Solis and Wets on local and global convergence as stated in reference [41] are:

**Theorem 1 (local search):** suppose that  $f$  is a measurable function,  $S$  is a measurable subset of  $R^n$  and  $H_1$  and  $H_3$  are satisfied. Let  $\{z_k\}_{k=0}^{\infty}$  be a sequence generated by the algorithm. Then,

$$\lim_{k \rightarrow \infty} P[z_k \in R_{opt}] = 1$$

where  $P[z_k \in R_{opt}]$  is the probability that at step  $k$ , the point  $z_k$  generated by the algorithm is in the optimality region,  $R_{opt}$ .

**Theorem 2 (global search):** suppose that  $f$  is a measurable function,  $S$  is a measurable subset of  $R^n$  and  $H_1$  and  $H_2$  are satisfied. Let  $\{z_k\}_{k=0}^{\infty}$  be a sequence generated by the algorithm. Then,

$$\lim_{k \rightarrow \infty} P[z_k \in R_{opt}] = 1$$

where  $P[z_k \in R_{opt}]$  is the probability that at step  $k$ , the point  $z_k$  generated by the algorithm is in the optimality region,  $R_{opt}$ .

In references [41], [44], and [45] The PSO was proven to straightforwardly comply with  $H_1$  as described below.

Let  $D$  be the function that describes the process of generating a new solution by PSO,  $h$  is the function performing PSO updates and  $f$  is the objective function (fitness). Assuming a unimodal minimization problem,  $D$  can be described as:

$$D(g_k, X_k) = \begin{cases} g_k & \text{if } f(g_k) \leq f(h(X_k)) \\ h(X_k) & \text{if } f(g_k) > f(h(X_k)) \end{cases}$$

By definition, it is clear that the function  $D$  is non-increasing. This means that the generated solution is always improving. Thus the PSO complies with  $H_1$ .

In order to prove (or disprove)  $H_2$  for PSO, one has to prove that areas explored by particles cover the entire search domain  $S$ . Investigations done in this direction by reported references proved that the union of sample spaces explored by all particles does not necessarily equal  $S$ . So, there is a nonzero probability that a part of the domain could be missed during the search. Accordingly the PSO fails to comply with  $H_2$ .

Violation of  $H_3$  was proved to be due to the possibility of having the PSO prematurely converge as  $X_j = l_j = g \ \forall j \in 1, \dots, M$ , where  $M$  is the population size [41]. In this case, the algorithm will stop making any further progress in the domain. Because  $g$  is not necessarily in or even close to the optimality region  $R_{opt}$ , the probability of sampling a point that is close to  $R_{opt}$  is definitely zero. This means that the hypothesis  $H_3$  is violated.

Therefore, the PSO fails to comply with  $H_2$  and  $H_3$ . According to theorems 1 and 2, the PSO is proven to be neither a local nor global optimizer. However, authors who studied this problem suggest that local and global search abilities can be guaranteed by manipulating the position equation to satisfy  $H_3$  and  $H_2$ . In references [44], [45] and [47], authors claimed that their proposed PSO variants were able to guarantee local and global optimality.

## 2.6 Diversity and PSO

As indicated earlier in this chapter, there are times when members of the swarm are dragged together to the global best position found so far by one of them. It is however, an expected consequence due to the nature of the PSO. The PSO is based on the cohesion

and alignment rules of Reynolds' model [1]. Besides, the roosting rule of Heppner's model is also adopted [48]. Therefore, the structure of the PSO promotes cluster formation and does not suggest any dispersing. What makes it even worse, is the use of the PSO with a global neighborhood topology. Thus, the neighborhood of a given particle is the whole swarm so that each individual has an access to the global best solution. Accordingly, during the course of iterations, individuals demonstrate a high tendency toward the best position found by one of them. There is always the possibility that this tendency might increase due to the fact that each particle seeks a point on the line connecting its personal best ( $l$ ) and the global best ( $g$ ). Therefore, it will converge (if there is any convergence) to a weighted mean of  $g$  and  $l$  as reported in section 2.5. Consequently, a position  $X$  can coincide with  $g$  and  $l$  causing a complete stagnation of the system. Although this is considered to be convergence from the *weak convergence* point of view (stability), there is no proof (or disproof) that this point will coincide or even be close enough to the optimal point. This phenomenon is called premature convergence. this is not unique to the PSO; any algorithm that performs population-based searches with stochastic moves can also suffer from this problem.

The logical solution for this problem is to diversify the population to avoid crowdedness. In the literature, many mechanisms are proposed to avoid the premature convergence by enhancing the global search ability of the PSO. It is clear that increasing the global search ability does not mean that the algorithm is definitely a global optimizer. Recalling  $H_2$  from section 2.5.4, the global optimizer is the algorithm that employs a search mechanism that must not repeatedly miss any sub-area inside the entire domain during the course of iterations. The word "any" in the previous verdict means that the probability of missing sub-area  $s$  inside the search domain  $S$  is zero. If the number of runs is considered in that verdict, the algorithm should locate the global optimum at each run in order to be called a global optimizer. For example, out of a hundred runs the algorithm should locate the global optimum a hundred times. Algorithm A that misses 30% out of a hundred is not a global optimizer but is better than the algorithm B that misses 50%. Mechanisms employed to diversify the population increase that percentage and do not necessarily address  $H_2$  directly for the PSO.

For example, Riget and Vesterstrom [49] proposed a so-called attractive and repulsive PSO (ARPSO). The ARPSO is a particle swarm optimizer that switches between two modes based on the value of a diversity sensor. In the attraction mode, particles update their position according to the original PSO update equations that already promote attraction. While in repulsion mode, the authors proposed a new update equation for the velocity by inverting signs of the cognition and social terms on the original velocity equation as:

$$V(t+1) = WV(t) - \phi_1(l(t) - X(t)) - \phi_2(g(t) - X(t))$$

During the run the algorithm frequently switched between the two phases according to the diversity measure:

$$diversity(S) = \frac{1}{|S||L|} \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^N (l_{ij} - \bar{l}_j)^2}$$

Where  $S$  is the swarm size,  $L$  is the length of longest diagonal in the search space,  $N$  is problem dimension,  $l$  is the personal best position and  $\bar{l}$  is the average value of  $l$ .

If the diversity measure is below a threshold  $d_{low}$ , the ARPSO will work in the repulsion mode and if it is higher than threshold  $d_{high}$ , the ARPSO will switch to the attraction mode. The authors were pleased with the results they achieved for a four multi-modal test problems due to the positively biased conclusion drawn from a comparison with the PSO and the genetic algorithm toward their technique. They also claimed that their mechanism helped to reduce the incidents of premature convergence to a large extent.

A very similar mechanism was proposed in [9] to add diversity to the PSO population using the self organized criticality (SOC) instead of the diversity measure reported earlier. The author stated that “the idea in the SOC that most state transitions in a component of a complex system only affects its neighborhood, but once in a while entire avalanches of propagating state transitions can lead to a major reconfiguration of the system.” In the SOC-PSO, each particle is given a variable  $C$  called criticality, which is initialized with zero value. Criticalities of individuals that are closer to each other than a pre-specified distance  $TD$  are increased by one each. Each individual is allowed to accumulate criticality up to a global criticality limit  $CL$ . If for a certain individual the criticality exceeded this limit, this individual has to relocate himself. Relocation of

individuals is associated with a so-called criticality dispersing that directly affects the criticalities of others in their neighborhoods and indirectly reflects on the whole swarm. The authors proposed two schemes for relocation: random re-initialization and pushing forward mechanisms. The results presented in the reference are far better than those obtained by the simple PSO.

Thanmaya et al. [50] introduced a new term to the velocity update equation to elevate the diversity level. Individuals in the swarm do not only learn from their own and global best experiences but also exploit the best experience of individuals in their neighborhood. The individual (other than the global best) that is selected to be exploited is the one that can maximize the fitness-distance ratio FDR:

$$\frac{fitness(l_j) - fitness(X_i)}{|l_j - X_i|}$$

where:

$l_j$  is the previous best position of the selected particle  $j$

$X_i$  is the position of the particle to be updated.

Therefore, the authors proposed an update equation in the form of:

$$V_i(t+1) = WV_i(t) + \varphi_1[l_i(t) - X(t)] + \varphi_2[g(t) - X_i(t)] + \varphi_3[l_j(t) - X_i(t)]$$

The authors found that, adding a third attractor to the particles has played a positive role in preventing premature convergence and in the performance of the PSO.

The concept of charged particles inspired by attraction and repulsion in electrostatics was introduced to avoid crowded zones in the population in reference [51] and reference [52]. Models of multiple swarms, with flow of information about fitness and distances among swarms have also been proposed to avoid premature convergence [53-55].

## **Chapter 3: New Formulation for the Swarm Optimization**

### **3.1 Introduction**

As stated earlier in chapter 1, a new stand is adopted to form the proposed swarm due to the lack of experimental evidences in Reynolds' distributed behavioral model [1] that is used to form the traditional PSO. In contrast, the proposed swarm optimizer is based on a well established theory in the field of sociobiology in order to create a more realistic-behaving swarm. In this chapter the balance between gregarious and social tolerance behaviors demonstrated by social animals is manipulated to form the proposed swarm optimizer. Accordingly, a set of lower level behaviors is defined to add the perspective of freewill to members of the swarm.

As researchers shift their efforts from seeking intelligence through complicated deterministic systems to complex adaptive ones, extensive work is going on in both mathematical modeling and ethological directions, to reveal the secret behind the success of social biological systems even in their very primitive forms. The complex adaptability features of these systems have enabled them to preserve their kinds for millions of years. One way to gain the most benefit from simulating the process of group decision making emerging by animal aggregations (in this context the word "swarm" and the expression "animal aggregation" are used alternatively) is to get a full understanding of these phenomenon and get an intimate look inside social and behavioral motivations. Most publications either in the field of ethology or mathematical modeling agree that swarms are formed by a balance between two social forces: attraction and repulsion. The concern of this chapter is the interpretation of animal behaviors that could lead to the formation of these forces. Also, understanding stimuli could definitely lead to downsizing the big

picture from a higher level phenomenon to “basic behaviors” that are easy to model. It is also important to explain the differences between complex adaptability and complicated determinability on systems in order to relate swarms and animal aggregations to systems that are complex and adaptively behaving.

### **3.2 Complex adaptive system**

The definition of a system in its broader sense is the collection of interacting elements where a change in the state of one element contributes to the whole group. Among systems themselves there are different types, for example a flock of birds and a car engine are both systems. However, a collection of marbles in a bag does not constitute a system simply because they are interacting [56].

#### **3.2.1 Adaptability vs. determinability**

The common feature between a car engine and a flock of birds as examples of two different types of systems is the way by which they are constituted (from a group of interacting elements). However, common sense can't relate these two examples to each other and no set or category, other than “system”, may have both examples under its definition due to their unlikeness. A swarm of birds with its reputation of having the most fascinating flocking behavior internally works in an unpredictable manner (each bird is responding differently to the change of his neighbor's state). However, the output of a car engine with different levels of input is known in advance. Not only the predictability, but also the relationship between input and output in both examples are different [56]. So, if one agrees to describe the relationship between the input and output in the car engine as “linear” since a small input yields a small proportional output and indeed a large output needs a large input, in that sense, flock of birds as a system has nonlinear relationship between its input and output. Unpredictability and nonlinearity are two common features that differentiate a flock of birds as adaptive system from the car engine as a deterministic system. In general, adaptive systems are constituted of elements that are usually called “agents”. Agents possess a concept of identity required as a ground belief



for having freewill to make decisions toward the level of response to changes on the surrounding environment.

### **3.2.2 Complexity vs. complicatedness**

A system is called complicated only if its performance completely depends upon both the health of its constituent elements and the strength of the connections among them. That is the system would fail to operate if one of its elements failed and it will work properly in a fully deterministic way if its elements do. For these systems, elements and their connections are both critical [56]. The car engine would not work properly if one of its components was not working or if the wiring was not fully connected. By contrast, a group of swarming birds would appear to us as a flock even if one of the birds failed to keep up. That is the feature of complexity on adaptive systems. Elements or agents themselves are not as critical as their connections [56]. The performance of the entire system does not completely depend on the individual behavior but rather on the degree of connectivity among agents. In addition, each agent responds unpredictably (but within certain rules) to changes inside the system. The other feature that distinguishes complex systems is a phenomenon called emergence. Emergent behavior is a type of behavior that is considered to be a source of novelty in a complex system. It is also described as the phenomenon that is responsible for the creation of fascinating high level behaviors from low level, non-strictly described interactions among agents. To summarize, complicated deterministic systems produce controllable and predictable outcomes, whereas complex adaptive systems can produce innovated and emergent outcomes [56].

## **3.3 Understanding swarms and animal aggregations**

### **3.3.1 Gregariousness**

One of the most realistic and sensible interpretations of flocking behavior of birds as an example of swarms is the work published in reference [57]. The author described the social attraction force of a member within the swarm to one of its kind as “gregariousness.” He defined the term “gregariousness” as the tendency of members to

respond positively to the presence of others of its kind. He also stated that “some psychologists describe it as a condition of responsiveness to social stimuli which, if blocked leads to frustration activities.” Others have described it as an appetite of hunger or sex and they tried to experimentally verify its remarkable consequence on isolated individuals [57, 58]. Struggling to catch others, stragglers from flock of starlings and passing crows responding to a group of their kind on the ground, were given as illustrative examples for gregarious behavior [57]. Although gregarious behavior among members of any animal aggregation is clear and needs no proof, scientists have found no hormonal demonstration for this behavior. Instead, this behavior was physiologically interpreted as a state in animals’ mind that probably leads to following a stereotyped neural pattern [57].

### **3.3.2 Social tolerance**

As stated earlier, forces that form and regulate animal aggregations are positive and negative interactions among members of the flock. While a positive interaction is demonstrated by gregariousness, negative interaction is established as a result of self expression, identity, or independence and is described as social tolerance [57]. In reference [57], the author described that factor as the force that is responsible for promoting animal aggregations by acting in opposition to the forces that bring animals together in response to gregarious behavior. He also preferred to describe that force in its negative aspect as he calls it social intolerance. The unstable situations that take place when a bird seeks a position that is too close to another settled bird on a section of telephone wire, was given by the author as an illustrative example of that force. A process of shuffling and reshuffling is then followed to keep at least six feet of spacing distance between each bird. The state of agitation that kept Craig’s birds [58] busy for hours to adjust both their appetite of getting close to a friendly mate and their aversion for crowding, is also given as an example. Unlike the force of attraction or gregariousness, social intolerance has hormonal origins. For example, injecting some birds with certain hormones (male sex or thyroxin) could lead to drastic effects on their social intolerability [59, 60].

### **3.3.3 Social forces and swarm's dynamism**

The interaction between gregariousness and social intolerance controls the process of forming and regulating swarm size, shape, and density. According to Emlen [57] in describing a flock of birds, “gregariousness initiates the process of aggregation and acts centripetally in drawing membership; while the social intolerance force serves as regulatory role by limiting the size of flock and preventing close crowding through its centrifugal action.” In his observations about cliff swallows; the author also mentioned that although the long telephone wires extended for thousands of feet, only few segments of the wire were used by birds to roost at any one time. At the beginning of aggregation; positive attraction force is clear as one or two birds start to land on the wire forming a nucleus for the flock. Shortly afterward, others randomly join the perching birds until aggregation of hundred or more accumulated all within the space of 100: 150 feet [57]. Birds first fill the central area of the flock and then the peripherals. Social intolerance forces were also demonstrated by members of swarm as shuffling and reshuffling occurred as needed and only when a bird landed too close to its neighbor. In conclusion, forces of attraction resulting from gregarious behaviors are responsible for initiating such aggregations. If left uncontrolled, one would see very compact clusters of these social animals. On the other hand, the negative force is a result of social intolerance or the expression of “self” to regulate the inner spaces among neighboring individuals. Speaking of the “self” and the identity in the case of non-human creatures will drag us to point out a fact that was denied for many years about the truth of cognition in animals. Contradictory to what most people believe, animals have some certain limits of free will and they can make decisions in their everyday lives, and surprisingly, these decisions are always “rational.” In the next few sections, animals’ freewill and the basic behaviors are described in more details.

### **3.4 Freewill in the animal world**

There is no doubt that the ultimate goal of any living organism is “survival.” Every living organism tries to keep its own internal body state (chemistry and

temperature) as constant as possible in the face of the changes in the surrounding environment. This phenomenon is called self regulation [61]. Self regulation is a feature of animate matters that frees them from the impact of environmental changes. Unlike inanimate matters, a living organism's state does not obey Newton's law but rather they tend to keep their own state constant no matter what forces are applied [61]. By definition, freedom and independence from the surrounding events is a sort of freewill associated with having an identity. This seems to be a common aspect among all animate matters ranging from cellular organisms to humans. For example, if you approach a bird you can guess that it will fly away but you can never be sure whether it will walk few steps or whether it will fly instantaneously as you get closer. In reference [61], the author presents examples for freewill in the insect world. He mentioned the example of ants choosing paths to their nest. Among an infinite number of paths, ants choose only a few to follow to their nest, and one can never predict which path will be followed. Moreover, differences in personality between two ants from the same colony are also demonstrated by choosing two different paths. It is very clear that freewill is a property of life and accordingly every living organism, no matter how primitive his brain enjoys certain level of freedom to make decisions or to choose a particular alternative out of many possibilities [61].

### **3.5 Proposed Basic behaviors**

Although it is very hard to determine basic behaviors from which animal aggregations are constituted, the earlier argument about identity and cognition is adopted to set low level behaviors that if combined together will result in high level ones. the gregariousness that is responsible for positive responses to others can be expressed among members of the swarm by imitation or copying behavior. On the other hand, social intolerance that is a reflection of "self" or "identity" can be expressed by the psychological state of every individual's mind inside the swarm. That psychological state is a combination of memory, thoughts, and emotions. To a certain extent, every individual tends to follow its own beliefs: a nature that could be described as "momentum" or the tendency to do particular actions regardless of whether these actions

proved to be successful or not. Also the concept of having free will is highly associated with decision making processes and randomness. Randomness in animal minds is expressed by play behavior. Finally, a rational decision making process is to be carried out by every individual to aggregate both gregariousness and social intolerance. So the set of basic behaviors that one is looking for can be expressed by:

- a- Imitation (as a symptom of gregarious behavior),
- b- Memory retrieval, play, and momentum (as reflections of “self” or social intolerance).

Figure 3.1 describes the process of combining a set of basic behaviors inside an agent’s brain to form his next move ( $X^{K+1}$ ), Where:

- $NbestX$ : represents the best successful experience in the neighborhood.
- $PbestX$ : represents agent’s successful experience.
- $X^K$ : agent’s current position.
- $RandX$ : random position acquired by the agent.

In Figure 3.1, the imitation represents the behavior that allows the agent to copy a part of the experience of the most successful agent in its neighborhood to its candidate move. The memory of the agent is a feature that brings up a part of the agent’s best successful experience to its candidate position. The play is a behavior that enables the agent to generate random changes to its candidate position. The momentum is the act of copying a part of the current position to the candidate one. The term “part” in the previous paragraph means that the agent decides according to decision indices the magnitude of adopted attitudes from others. More details about decision making and decision indices are given in the coming chapters.

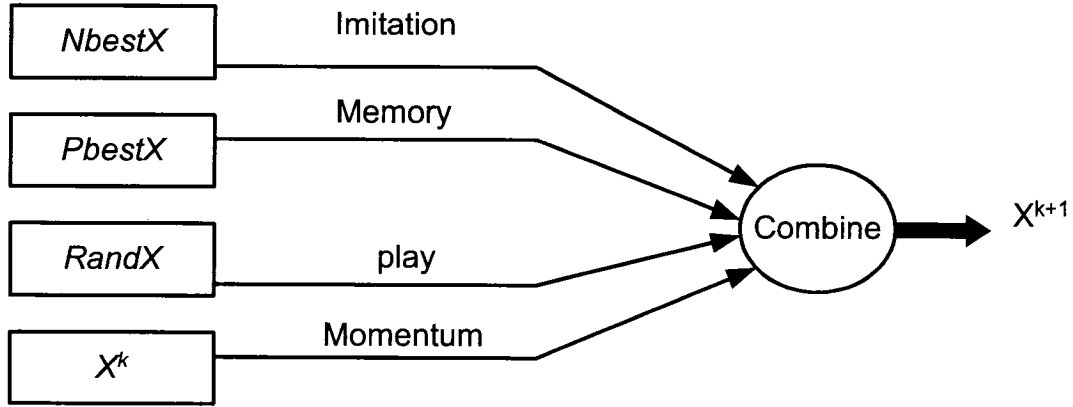


Figure 3.1: Combining basic behaviors to form a candidate move.

In the next few sections, basic behaviors in the social animals' world are given in greater detail.

### 3.5.1 Evidence of memory retrieval

Scientists used to think of birds in a more robotic way, denying their ability as cognitive creatures. The behavior of birds returning to places where they hide their food was always interpreted as a simple act of returning to preferred places where they would happen onto their caches [62]. Safe arrival of migrating birds to their destination was attributed only to their internal compass setting, guided by the earth's magnetic field and using a finely tuned sense of smell [63]. However, recent research work investigated memory and skills of mental time travel in birds [64, 65, and 66]. Episodic-like memory and whether birds can remember *what*, *where*, and *when* is also researched [67, 68, and 69]. Surprisingly, investigations in this direction have revealed that navigational skills (used during migration) in older birds is more sophisticated than those in younger birds, which proves their ability to learn from past experience. In laboratory tests, birds demonstrate a remarkable level of recalling locations of food they buried months earlier and can even retrieve favorite foods first [65]. Although there is no evidence that animals can plan for their future or even feel pain when remembering sad events from their past

(the way humans do), an extensive amount of research work is going on in this area for the welfare of these creatures.

### **3.5.2 Imitation**

Imitative behavior among animals is quite clear and needs no proof even in a heterogeneous environment where animals are trained to imitate human acts. Acquiring new skills by imitation is the most common mechanism of learning among animals. Laboratory experiments proved that young rats acquired a new feeding technique of stripping pine cones to get the nutritious seeds inside by watching their experienced mothers doing it [70]. On the other hand, adult rats that have never been previously exposed to such an experience were not able to do it efficiently. Moreover, high level human-like acts gained by imitation were also pointed out in reference [71] among animals. There are different simple mechanisms that could appear to observers as imitative behaviors among animals. One of these mechanisms is the contagious behavior that could be simplified by the rule of “if others are fleeing, flee also” [72, 73]. In describing this mechanism, the author of reference [71] stated that: “the idea is that the stimuli produced by the performance of a particular behavior serve as triggers for others to behave in the same way.” Possible examples of contagious behavior include flight responses, movement in flocks or schools, and chorusing by birds [71]. It is also worth mentioning that a contagious behavior does not deny the degree of freewill that every living organism enjoys. A person may yawn if a nearby person does, yet the former can still have his own will which is known to all of us to be “free.”

So, it must be something that is transmitted socially (rather than genetically) that enables animals to solve a complex correspondence problem in translating novel visual input from observation into appropriate motor output [71].

### **3.5.3 Play**

Being a living organism any animal enjoys a certain degree of personality or identity which allows some degree of freedom in order to be able come up with some unpredictable acts. This is also a common feature that is required for an element of any

system to be called an “agent.” In the animal kingdom, unpredictability or randomness can be specified by play behavior. There is no immediate benefit of play behavior for animals. On the contrary, play behavior could put an animal at risk due to the potential of being attacked by predators while busy playing.

### **3.6 Summary**

Although simple rules could lead to lifelike patterns, there is no guarantee that interactions among individuals of a swarm or aggregations (of living organisms) are that “simple.” Cognition-related issues such as identity, freewill, and decision making are discussed. After all, it is hard to believe that interactions among heterogeneous (from the sense of different personalities) individuals when carrying out a collective task (for example flocking) in the same space and at the same moment of time, would be from the type that could be called “simple.” The term “simple” shouldn’t describe “interactions” themselves but rather describes the framework that governs responses due to these interactions. Given flocking behavior, every bird is connected to the flock by a certain membership of belonging and has to respond to others in a way that satisfies simple rules; preventing compact crowding and allowing harmony of flying. That is also a good reason to rule out the idea of birds having crisp thinking that relies on the sense of accurate calculations of distances and velocities. Instead, it would be more realistic to model bird’s cognitive minds during flocking by fuzzy reasoning.



## **Chapter 4: The Proposed Technique**

### **4.1 Introduction**

In this chapter, a model of the fuzzy reasoning process is proposed for each member (agent) inside the swarm. Agents are assumed to be rational where the decision making process is to be made on a rational basis. Agents have to make up their mind in accordance with a multi criteria decision making process. Precisely, agents have to make their decisions based on how much they are responding to others. In other words, they have to make a decision that reflects a trade-off between imitating others' behaviors (gregariousness) and retrieving their own past experiences (as an aspect of social intolerance). Imitation is simply copying others' success. The process of recalling previous experiences is the memory retrieval part of the cognitive process. Also, the momentum and play behaviors are considered in the decision making process and consequently on the genotype representation of agents.

For population-based search methods, promoting high fitness and maintaining diversity of the population is the ultimate goal for people working in this area. The issue of fitness-diversity correlation in genetic algorithms has been extensively discussed in the literature. However, for the particle swarm optimizer this issue is not fully addressed. In

this chapter, the decision making process is introduced. Decisions are made to promote high fitness and elevate the diversity level to prevent premature convergence. The Levenshtien edit distance method is used to measure the distance between two individuals in the genotypic space. The so-called Yager's Ordered Weighted Average (OWA) is employed to aggregate decision criteria [74].

#### 4.2 Multi criterion decision making (MCDM)

Decision making is a process of choosing one alternative among a number of possible alternatives by an agent. That alternative has to be characterized by a form of rationality and fits the decision maker's goals and desires. It is also very important to distinguish between decision making based on a single criterion and those involving multiple criteria. Single criterion decision making is simply an optimization problem to find the best alternative that meets constraints or requirements, and is based on a single objective like profit or cost function. However, in multiple criteria decision making, one has a finite number of usually conflicting criteria that influences the choice of one alternative over others. A decision maker has to aggregate these criteria by means of a suitable aggregation operator in order to reach a compromise to guide the choice of one of the available alternatives. In the proposed technique, the multi-criterion decision making is used due to the availability of two criteria (fitness and diversity) for every agent. In reference [75] the author summarizes the problem of MCDM as the problem of choosing among  $n$  alternative that is based on  $m$  criteria. He also denotes the available alternatives as  $A_1, \dots, A_n$  while the criteria are indicated by  $C_1, \dots, C_m$ . For a typical MCDM there is a decision table (Table 4.1) that provides a concise way of arranging the available information and can be utilized by a broad range of decision making algorithms [75]. In Table 4.1 each row represents an alternative and each column belongs to a criterion. Scores  $S_{ij}$  describe the performance of alternative  $A_i$  against criterion  $C_j$  where  $i=1, \dots, n$  and  $j=1, \dots, m$ . Weights  $W$ 's are then assigned to each criterion by the decision maker in a skewed way to reflect the importance of each criterion on the performance of alternatives. Multi criterion decision making methodology is then applied to identify one alternative or at least a short list of admissible alternatives.

Table 4.1: Decision table [75]

	$C_1$	.	.	$C_m$
	$W_1$	.	.	$W_m$
$A_1$	$S_{11}$	.	.	$S_{1m}$
.	.	.	.	.
.	.	.	.	.
$A_n$	$S_{n1}$	.	.	$S_{nm}$

Generally, two procedures are used in multi-criteria evaluations and are outlined in reference [5] as classical crisp and continuous fuzzy procedures.

#### 4.2.1 Crisp reasoning based procedure

Traditionally there are two methods to reach a crisp supported decision. First, the criteria are expressed by Boolean values as “TRUE” and “FALSE” or “YES” and “NO” and then an averaging binary operator such as AND (intersection) or OR (union) logic are used for aggregation. Although the technique is very straightforward it does possess major drawbacks. One known drawback is that wherever you draw boundaries between the set of inadmissible criteria and those that are admissible, there is always a discontinuity due to the fact that a drastic change in output might take place as a consequence of insignificant incremental changes in the input. Accordingly, an alternative that is relatively acceptable could end up rejected due to this type of judgment. For example, the intersection operator may represent an aggressive case i.e. an alternative which is ideal according to  $m-1$  criteria will be removed from consideration if it fails to fulfill just one of its requirements. On the contrary, the union operation is far too liberal, where it is sufficient if the alternative meets only one of its criteria, regardless of how bad the values of the remaining ones are [5].

Another widely used MCDM technique is based on summations of normalized weighted values of the criteria known as Simple Additive Weighting (SAW) or the

Weighted Linear Combination (WLC) method. In this technique continuous criteria are first rescaled to a common evaluation scale (normalized) and then averaged according to assigned weights of importance as follows [5]:

$$S_{wlc} = \sum W C_n / \sum W$$

where

$S_{wlc}$  Selection index

$W$  Weight of importance

$C_n$  Normalized criterion

This technique might overcome some of the drawbacks of the previous method by having a continuous measure for the selection index to prevent extreme risk situations (risk aversion by AND operator and risk taking by OR operator) and avoid rigidity of borders between acceptance and rejection. However, reaching a decision using a rationale of just accumulating subjectively weighted criteria is not widely acceptable.

#### 4.2.2 Fuzzy reasoning based procedure

This procedure is fairly related to Weighted Linear Combination, but is capable of generating a vast range of decision strategies [76]. This approach attempts to transform the synthetically crisp criteria of the Boolean approach into continuous criteria that express a degree of aptness by modeling criteria using continuous variables ranging from most appropriate (value 1) to least appropriate (value 0) [77]. Most of the drawbacks mentioned in the previous subsection can be avoided by strengthening the utilized reasoning using fuzzy measures and fuzzy operations. In this approach alternatives are expressed by a fuzzy set where a small change in the input will just produce a small change in the membership grade to avoid abrupt jumps between extremes [5]. Moreover, using fuzzy t-norms and t-conorms instead of the crisp intersection and union operators can also be more beneficial, especially if suitability evaluation is reached by means of a fuzzy averaging operator. It is always good to reach a decision that permits trade-off between criteria i.e. a good value in one variable may compensate for a bad value in another. In that sense, the aggregation operator can be seen as AND/OR-operator which allows full trade-off between all criteria. A solution to this issue is proposed by Yager [74]

who presented a method called Ordered Weighted Average (OWA) with continuous control over degree of ANDOR-ness and with independent control over the degree of trade-off [5].

### 4.3 Elements of the decision making process

Decision making is the study of identifying alternatives based on the values and preferences of the decision maker [75]. The process consists of a set of essential elements:

1. Goals
2. Alternatives
3. Criteria
4. Decision making rule

The next subsections these elements are explained in more detail.

#### 4.3.1 Goals

Traditionally, every agent (individual) in the swarm updates his move in accordance with the highest fitness individuals in domain space. The proposed technique adopts new criteria in adjusting the moves of every individual. The diversity of population along with the fitness are taken into consideration. The diversity of population in population-based optimization methodologies has been a significant issue. Promoting a solution for only high fitness could be extremely tricky due to the high possibility of getting a premature convergence as a result of tracking only good solutions and avoiding other areas in the problem domain.

As a result, the goal is set to update agents' moves in accordance with promoting both fitness and diversity in the population. The proposed technique redirects the selection of solution from simply the fittest to the fittest and the most diverse. Accordingly, in their decision making process, agents prefer alternatives that are both fit and diverse.

### 4.3.2 Alternatives

The proposed technique is based on simulating a decision making process within every agent's mind to rank a finite set of available alternatives. According to the defined goal for the MCDM process, these alternatives are:

- 1) Imitation Alternative: Imitating the success of swarm mates.
- 2) Retrieval Alternative: Retrieving memories about one's own successful experience.
- 3) Momentum Alternative: Continuation in the same direction.
- 4) Play Alternative: Adopting some random moves.

Individuals use single evaluation function (decision rule) to rank the alternatives and then determine the magnitude of genotypic parts to be copied into their next move.

### 4.3.3 Criteria

The basis for a decision making process is the criteria that form an objective measure of the goal and are able to measure how well each alternative achieves the goal [75]. Criteria that are used to achieve the specified goal (promoting fitness and diversity) are:

- 1- Fitness
- 2- Levenshtien edit Distance (as measure of diversity)

More details are given below for fitness and distance as two criteria for the proposed MCDM.

#### 4.3.3.1 Fitness

Fitness is determined by evaluating the problem's objective function for every agent at every move. For imitation alternative; the fitness is the best objective function value in the neighborhood whereas, for retrieval alternative; the fitness is the value of the objective function for the individual's best experience. For the play and momentum behaviors, fitness is described by the value of the objective function for a random permutation and the current position, respectively.

#### 4.3.3.2 Levenshtien edit distance

As mentioned earlier, the diversity of a population is an important factor to reach the global optimum of a problem. Loss of diversity could lead the swarm to act as parallel separate hill climbers instead of cooperative agents. Therefore, incorporating the diversity of the population in the position updates is a very novel step in the swarm optimization and will cause substantial advantages due to the anticipated resistance to premature convergence. Incorporating diversity along with fitness in updating agents' moves benefits PSO by adding more common sense through the transformation of calculations from subjective to objective.

Levenshtien edit distance (LD) [18] is traditionally used to measure the distance between two strings or sequences to determine their similarity. The method is named after the Russian scientist Vladimir Levenshtien who devised the algorithm in 1965 [78]. In the proposed technique, permutation encoding is used to represent agents in the genotype space. Therefore, Levenshtien edit distance is found to be suitable for measuring the distance between two agents in the proposed swarm.

LD distance is defined as the number of deletions, insertions, or substitutions required for transforming the permutation  $s$  into permutation  $t$  [78]. The greater the difference between the two permutations, the greater the LD distance is. For example, the LD between two permutations  $t : \{ 3 \ 2 \ 1 \ 4 \}$  and  $s : \{ 3 \ 2 \ 1 \ 4 \}$  is  $LD(s,t)=0$  (two permutations are identical). The LD between two permutations  $t : \{ 3 \ 2 \ 1 \ 4 \}$  and  $s : \{ 3 \ 1 \ 2 \ 4 \}$  is  $LD(s,t)=2$ , because two changes are required to transform  $s$  into  $t$ .

Steps to calculate LD between two permutations  $s$  and  $t$ , can be summarized as follows [78]:

Step 1:

*Set  $n$  to be the length of  $s$*

*Set  $m$  to be the length of  $t$*

*If  $n=0$ , return  $m$  and exit*

*If  $m=0$ , return  $n$  and exit*

*Construct a matrix containing  $0....m$  rows and  $0....n$  columns*

Step 2:

*Initialize the first row to  $0....n$*

Initialize the first column to 0....m

Step 3:

Examine each element of  $s$  ( $i$  from 1 to  $n$ )

Step 4:

Examine each element of  $t$  ( $j$  from 1 to  $m$ )

Step 5:

If  $s[i]$  equals  $t[j]$ , the cost is 0

If  $s[i]$  doesn't equal  $t[j]$ , the cost is 1

Step 6:

Set cell  $d[i,j]$  of the matrix equal to the minimum of:

- the cell immediately above plus 1:  $d[i-1,j]+1$
- the cell immediately to the left plus 1:  $d[i,j-1]+1$
- the cell diagonally above and to the left plus the cost:  $d[i-1,j-1]+cost$

Step 7:

After the iteration steps (3,4,5,6) are complete, the distance is found as the cell  $d[n,m]$ .

Example: Find the edit distance LD between two agents in genotype space where their positions are expressed by the two permutations  $t$  and  $s$ .

Where:  $t : \{ 3 \ 2 \ 1 \ 4 \}$  and  $s : \{ 3 \ 1 \ 2 \ 4 \}$ .

Solution: The two permutation  $t$  and  $s$  are arranged as shown in the Table 4.2. Values in the cells are computed according to the LD algorithm steps.

Table 4.2: Calculation of LD between permutations  $t$  and  $s$ .

		$t \rightarrow$				
		<b>3</b>	<b>2</b>	<b>1</b>	<b>4</b>	
$s \downarrow$	0	1	2	3	4	
	<b>3</b>	1	0	1	2	3
	<b>1</b>	2	1	1	1	2
	<b>2</b>	3	2	1	2	2
	<b>4</b>	4	3	2	2	<b>2</b>



The Levenshtien distance LD is the value in the lower right hand corner of Table 4.2 and equals 2.

#### 4.3.3.3 Criteria Fuzzification

Another important step in the MCDM process is the standardization of the criteria. In order to reach a final aggregation score, the criteria are transformed into comparable scales measured according to a standardized range. The process of fuzzification in a fuzzy set theory is employed to standardize the criteria for the final aggregation function. The fuzzification process transforms any value of criterion to a normalized value in the range of  $(0-1)$ . This value expresses a fuzzy membership grade and indicates a continuous increase from non membership to complete membership [79]. A significant issue in the fuzzification of criteria is the choice of limits for the membership grade. These limits determine the lower and the upper boundaries for acceptable values of each criterion.

To choose fuzzification limits, a cost minimization function is considered. According to the goals; one wants to promote higher fitness and maintain diversity. In that sense, membership values are calculated. For the fitness criterion, the higher the fitness, the higher the membership grades are. Memberships are therefore specified by a decreasing function where the lowest membership grade is given to the highest value of the objective function and the highest membership grade is given to lowest value of the objective function in the phenotype space.

For diversity (distance) criteria, the further the distance, the higher the membership grades. In this case, memberships are specified by an increasing function. Limits for the fuzzification process are given in Table 4.3.

**Table 4.3: Limits on the fitness and the distance for fuzzification process.**

Criterion	Limit1	Limit2
Fitness	(Lowest value of objective function)	(Highest value of the objective function)
Distance	Lowest value of distance min(LD)	Highest value of distance max(LD)

Membership functions used for calculating fuzzy sets of the two criteria are shown in the Figure 4.1.

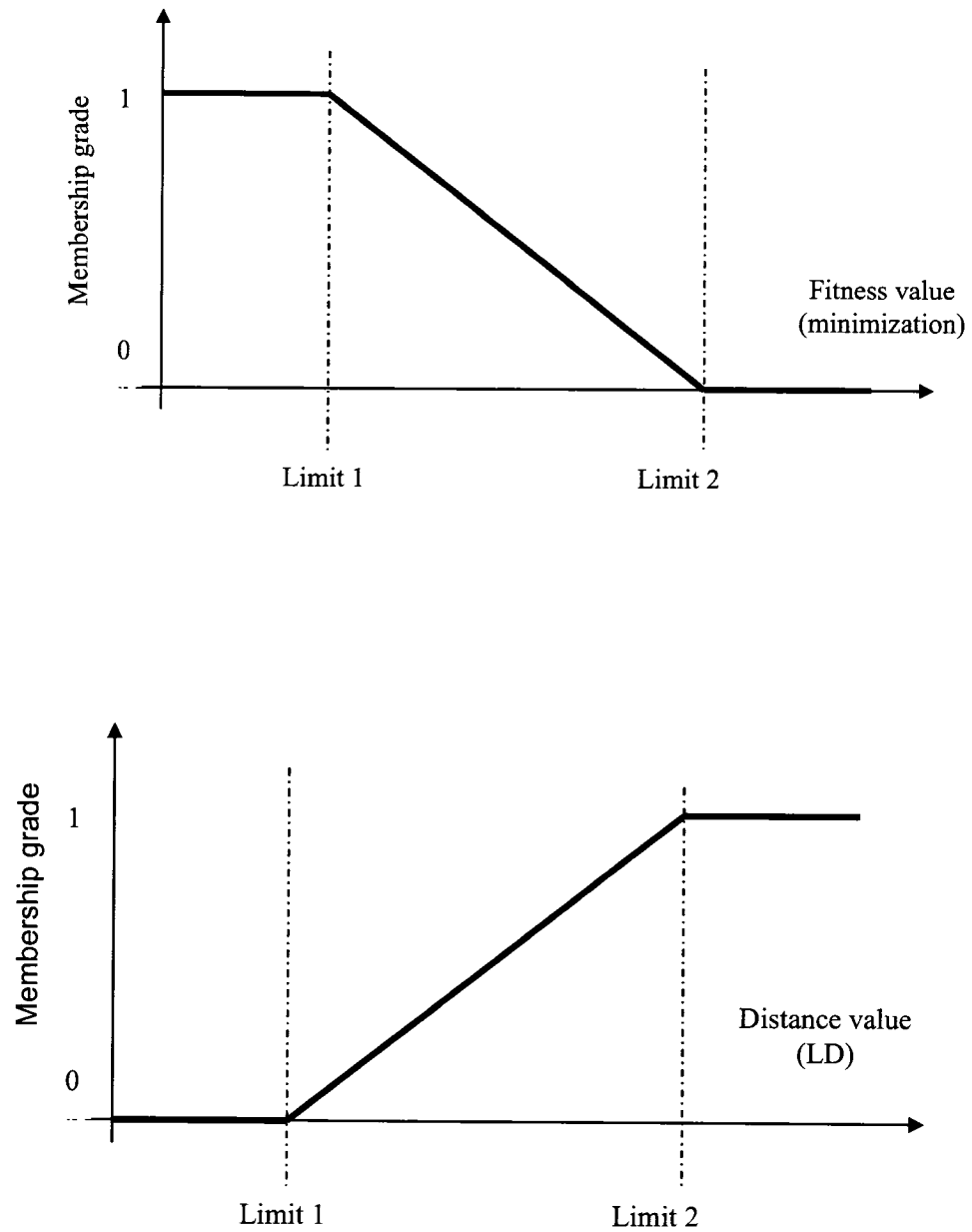


Figure 4.1: Membership function used in calculating fuzzy sets of the criteria

#### 4.3.4 Decision rule: fuzzy OWA aggregation operator

The final and most important step in the MCDM process is forming an overall function that aggregates the criteria. At one extreme, at least one of the criteria has to be satisfied. This represents a pure OR operation. This operator is known to be the risk taking alternative in the decision making process. At the other extreme, one needs all the criteria to be satisfied. This represents the pure AND operation. This situation is known as risk averse. To avoid these two extremes, Yager [74] introduced a new averaging operator called the ordered weighted average (OWA). This operator provides an aggregation that provides a different degree of ANDORness and lies in between those two extremes. By choosing appropriate values for the so-called OWA weights, one can model any degree of ANDORness between 0 (pure AND) and 1 (pure OR). In fact, the OWA operator extends the space of quantifiers from the *pair* {for all ( $\forall$ ), at least one ( $\exists$ )} to the *interval* [for all ( $\forall$ ), at least one ( $\exists$ )] [80]. Using the OWA procedure for MCDM results in decision strategies that vary along two dimensions: risk and tradeoff [76]. Risk levels are determined by the degree of ANDORness expressed by the weights of the OWA operator. The tradeoff level is determined by the value of dispersion or the skew in the weights of the OWA. Figure 4.2, which is reproduced from references [76] and [80], shows the OWA in the perspective of risk-tradeoff space of fuzzy aggregation operators.

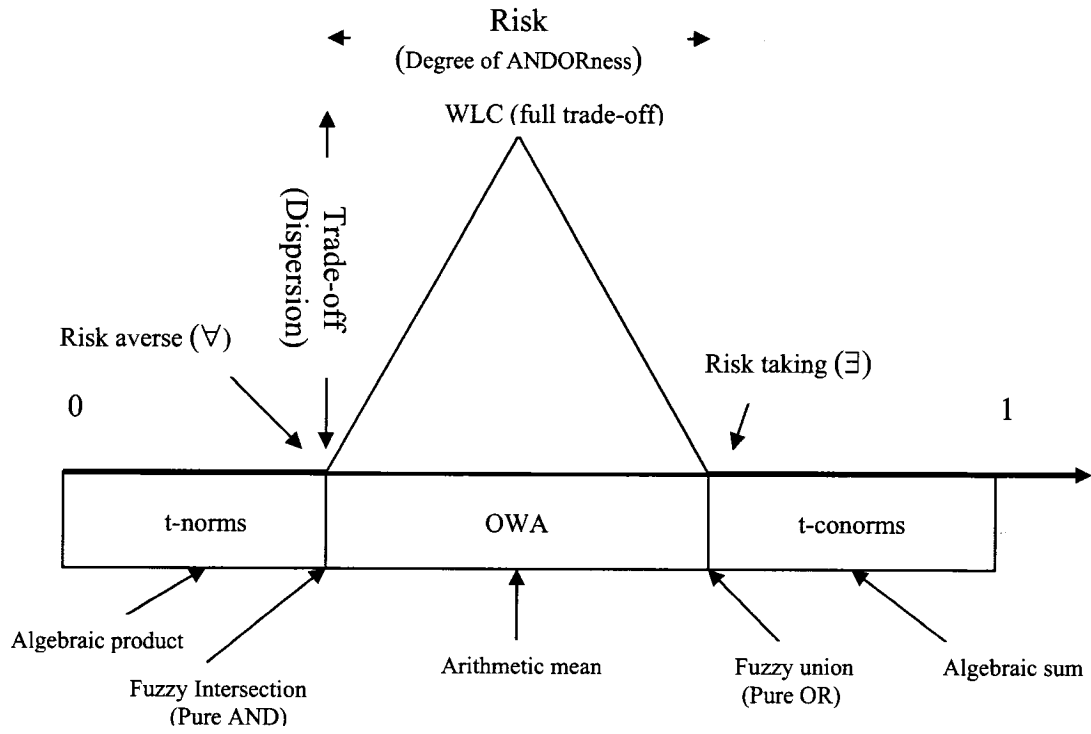


Figure 4.2: The OWA in the perspective of risk-tradeoff space of fuzzy aggregation operators

According to Yager [74], the OWA is defined as a mapping  $F$  from  $I^n \rightarrow I$  (where  $I = [0,1]$ ) which is called an OWA operator of dimension  $n$  if associated with  $F$ , there is a weighting vector  $W$ ,

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix} \text{ such that}$$

$$1- W_i \in (0,1)$$

$$2- \sum_i W_i = 1$$

Where  $F(a_1, a_2, \dots, a_n) = W_1 b_1 + W_2 b_2 + \dots + W_n b_n$

Where  $b_i$  is the  $i^{\text{th}}$  largest element in the collection  $(a_1, a_2, \dots, a_n)$ . Yager called an  $n$  vector  $B$  an ordered argument vector if each element  $b_i \in [0,1]$  and  $b_i \geq b_j$  if  $j > i$ .

Given an OWA operator  $F$  with weight vector  $W$  and an argument tuple  $(a_1, a_2, \dots, a_n)$  one can associate an ordered input vector  $B$  with this tuple such that  $B$  is the vector consisting of the argument of  $F$  put in descending order. Using this notation then  $F(a_1, a_2, \dots, a_n) = W'B$ ; the inner product of  $W$  and  $B$ . Yager also emphasized that the weights are associated with a particular ordered position rather than a particular element. For any ordered vector argument vector  $B$  and any OWA operator  $F$  with weighting vector  $W$ :

$$0 \leq F(B) \leq 1$$

An outline of an MCDM supported by ordered weighted average (OWA) can be summarized as shown in Figure 4.3 [5].

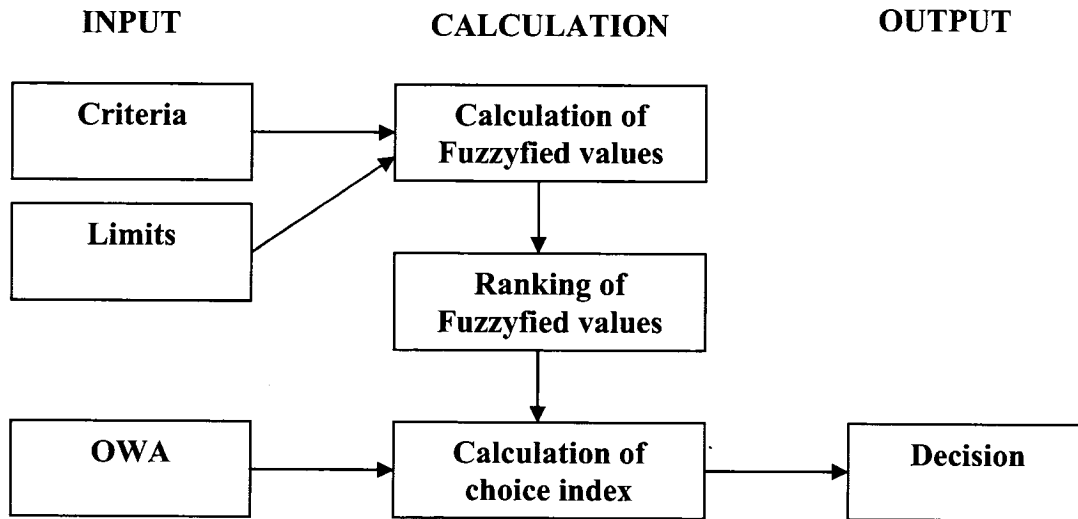


Figure 4.3: OWA-supported decision making [5]

#### 4.3.5 Agents' position update

Every agent in the swarm performs the decision making process to adjust its next move in the space. Define the following symbols:

$X^K$  : Position at iteration  $K$

$X^{K+1}$  : Candidate position at iteration  $k+1$

$gbestx$  : Position due to global best fitness

$pbestx$  : Position due to personal best fitness in agent memory

$Nbestx$  : Position of agent that has the best fitness in the neighborhood

$Randx$  : Randomly generated position

$gbest$  : Global best fitness in the domain

$pbest$  : Personal best fitness in agent memory

$Nbest$  : Best fitness value in an agent's neighborhood

$Rndft$  : Fitness due to random position

$Xft$  : Fitness at current position  $X^K$

Every agent ranks the four available alternatives in accordance to their performance with the two criteria. Alternatives in terms of criteria (fitness and distance) can be arranged as shown in Table 4.4:

**Table 4.4: Alternatives vs. criteria.**

	Fitness	distance
Alternative 1	$Nbest$	$LD(Nbestx, X^K)$
Alternative 2	$pbest$	$LD(pbestx, X^K)$
Alternative 3	$Xft$	$LD(X^K, X^K)$
Alternative 4	$Rndft$	$LD(Randx, X^K)$

Each criterion is then fuzzified according to their control points (limits). Fuzzy membership values are then ranked in descending order as shown in Table 4.5.

**Table 4.5: Ranked fuzzified (RF) values for each alternative**

	$RF_1$	$RF_2$
Alternative 1	$A_{11}$	$A_{12}$
Alternative 2	$A_{21}$	$A_{22}$
Alternative 3	$A_{31}$	$A_{32}$
Alternative 4	$A_{41}$	$A_{42}$

Where values of  $RF_1$  are greater than values of  $RF_2$ .

Having obtained the ranked fuzzy values for each alternative, the values of order weights associated with OWA need to be specified. The values of order weights are determined according to the decision maker preferences and based on the desired level of risk and tradeoff between criteria. For example, order weights of  $(1, 0)$  will give full weight to the highest  $RF$  of each alternative regardless of how poorly ranked the other criterion. This strategy is represented by the risk taking or optimistic decision making in the risk-tradeoff space described earlier. However, order weights of  $(0, 1)$  will give full weight to the poorest  $RF$  no matter how good the other criteria. This decision strategy is described in risk-tradeoff space by risk averse or pessimistic decisions. In simulations, the equation  $W_k = (k/n)^\alpha - ((k-1)/n)^\alpha$  suggested by Yager is used to calculate the order weights.

Where;

$k=1, \dots, n$

$n$ : number of criterion

$\alpha$ : A parameter allows the transformation from a qualitative scale to an order's weight [81].

Finally, an overall ranking index or choice index is calculated for each alternative as:

$$\text{Ranking index (RI)} = \sum RF_k * W_k .$$

The ranking indices for each alternative are then used to determine the magnitude of imitation and memory retrieval from the corresponding position permutations. The candidate position of an agent is based on number of elements that will be copied by imitation and memory retrieval from the corresponding permutations into the permutation of the candidate position.

### Summary

This chapter, the manipulation of the gregarious and social intolerance behaviors in the animal world to form the swarm is presented. The proposed decision making process carried out by the swarm agents is also described. Details of the position update by means of the set of basic behaviors are given. The decision strategies including goals, alternatives, and criteria associated with the process are described. The fuzzy aggregation function OWA and its weights are also demonstrated.

# Chapter 5: Applications

## 5.1 Introduction

In the previous chapter a description of the simulated decision making process was given. Decision criteria and fuzzy aggregation were also described. In this chapter the performance of the proposed optimizer is examined. Two applications are considered here: the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). First, the technique is tested using a small size TSP problem to show the procedure, followed by an agent to reach a decision. The technique is then tested with larger size TSP problems (Symmetrical and Asymmetrical). In addition, the technique is extensively tested using the quadratic assignment problem. Over a hundred problems from different applications are considered in the simulations. A comparison between the proposed technique and the standard genetic algorithm is also performed. Moreover, a comparison between the performance of the technique and the traditional PSO is carried out for some problems.

## 5.2 The Traveling Salesman Problem

Due to its generality, the traveling salesman problem is widely used as a benchmark for testing new classes of combinatorial optimization algorithms.



### 5.2.1 Background

In TSP, a salesman has to make a round trip between a given number of cities. In the tour, the salesman has to minimize the traveling cost of the overall tour. He also has to visit each city exactly once. The history of the problem dates back to the mid 1700's when Euler made his first attempt to find the optimal tour of a knight over the 64 squares of chessboard [82]. The origin of the TSP as known today has many conflicting stories. However, among the research community [83-87] the most credible story is stated as follows.

The first nonmathematical formulation of the problem was introduced by a successful German salesman in 1832. The Austrian mathematician K. Menger is thought to be the first to introduce a mathematical statement of the problem in 1928. During his visit to Harvard between 1930 and 1931, Menger exposed the problem to American scientists. At that time Hassler Whitney was a PhD student at Harvard working on graph theory and it seems that some discussion on this matter had taken place between Menger and Whitney. Almost one year later, Whitney was working with the National Research Council at Princeton and is believed to have given a seminar on a 48-states TSP. In 1937 Merrill Flood was attempting to obtain a near optimal route for a school bus problem. He mentioned that he got some ideas for his routing problem from W. Tucker who heard about the TSP from Whitney during his studies at Princeton University. In 1948 Flood was encouraged by RAND (Research AND Development) corporation of Santa Monica, California to popularize the TSP. In 1954 the first algorithmic attempt toward solving the TSP was introduced by RAND researchers G. Dantzig, R. Fulkerson, and S. Johnson [85].

Since that time the TSP attracted researchers' attention due to its challenging nature. The problem is easy to formulate but yet hard to solve. The only way to find an exact optimal tour in a TSP can be done by using complete enumeration of all possible tours in the search space, a process that can grow dramatically with size of the problem. For example, the problem of 20 cities requires evaluations of factorial of 20 ( $2.4329 \times 10^{18}$  possible tours). Assuming a computational time of  $1 \times 10^{-9}$  seconds for every tour, a total computer time of 77.1468 years is required for complete enumeration.

Heuristic search techniques have proven capable of finding near optimal solutions for such problems by evaluating only a very small fraction of all possible tours. The swarm

optimizer is one of the successful heuristics that have been recently proposed and rapidly promoted in different research areas.

### 5.2.2 TSP and graph representation

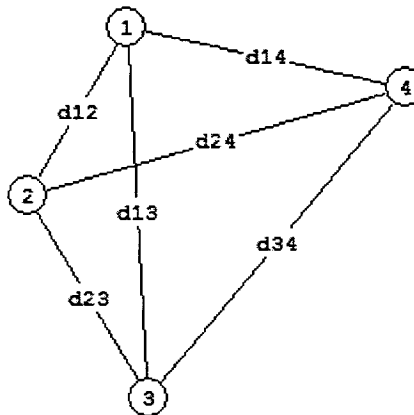
In graph theory, the TSP can be expressed by a weighted-edge graph  $G: (V, E, D)$  where:

$V$ : set of vertices (cities)

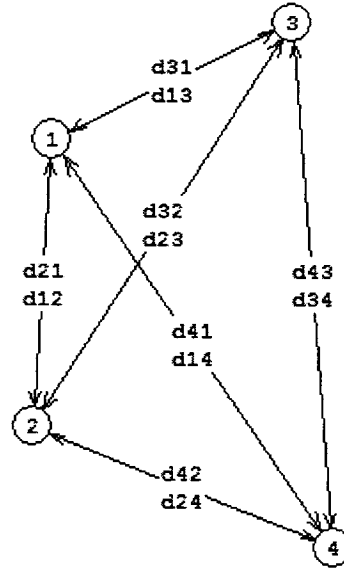
$E$ : set of edges (roads connecting two cities)

$D$ : weights given to each edge (distances or cost of traveling). There are two well-known categories of the TSP; namely the symmetrical and asymmetrical traveling salesman problems (STSP and ATSP respectively).

In graph theory terminology, the STSP is expressed by a weighted-edge undirected graph whereas the ASTP can be expressed by a weighted-edge directed graph. Figure 5.1 shows these two types (the graph is created using Graph Interface (GRIN) software [88]).



(a) Undirected graph



(b) Directed graph

**Figure 5.1: Symmetrical and Asymmetrical TSP representation.**

In the case of STSP the weights of edges connecting two given cities in both directions (forward and backward) are equal. It means that the cost of traveling from city  $i$  to city  $j$  is equal to the cost of traveling from city  $j$  to city  $i$  ( $d_{ij}=d_{ji}$ ). However, in the case of ATSP this is not applicable i.e. ( $d_{ij}\neq d_{ji}$ ).

Generally, to solve the TSP problem one must find the Hamiltonian cycle that has the least cost of traveling or the least distance. A Hamiltonian cycle is a complete tour among all cities ending at the starting point, providing each city was visited only once and edges were not repeated. In the simulation, each generated Hamiltonian tour in the studied problem is coded as an ordered permutation  $\pi$ . Mathematically, the problem can be stated as:

$$\text{minimize } \sum_{i=1}^{N-1} d_{\pi(i,i+1)} + d_{\pi(N,1)} \quad (5-1)$$

Where:

$N$  is the number of total cities;

$\pi : A \leftrightarrow B$  is a bijective (one-to-one and onto) function from a set A to a set B that defines a permutation representing a given Hamiltonian tour;

$d_{\pi(i,i+1)}$  : The distance or the cost of traveling between a city that located in the order  $i$  and a city located in the order  $i+1$  in the permutation  $\pi$ ;

$d_{\pi(N,1)}$  : represents the cyclic part of the tour.

### 5.2.3 Demonstrative example

In this section, the ATSP problem of 17 cities (br17) [90] is given as an illustrative example to explain in detail the major steps of the proposed technique. Data, cost (or distance) matrices, and optimal tours for the TSP problems presented here can be found in the online library of the TSP problem (TSPLIB) [90].

In the simulation, a swarm size of 30 is employed to solve an asymmetrical 17 city problem. The entire population is divided into five equal neighborhoods based on orders of agents. For a given instant of time  $k$ , the attributes of the algorithm are captured in Figure 5.2.

An agent that is characterized by the position  $X^k$  and the fitness value of 235 is located within the neighborhood N.

		Current positions for a specified neighborhood																	
Best in the neighborhood →	Fitness	74	14	1	12	11	15	16	9	13	10	7	8	17	5	4	6	2	3
		75	12	3	14	10	6	16	7	13	9	1	15	4	5	8	11	2	17
		245	5	3	12	9	1	16	7	13	10	15	4	17	14	8	6	2	11
An Agent X →		235	13	3	8	9	15	2	6	14	1	12	7	4	16	17	11	10	5
		229	14	3	6	1	9	10	7	13	5	17	8	4	16	12	2	11	15
		240	5	9	12	6	15	13	14	16	7	17	4	1	10	8	11	2	3
pbest		Best previous position for an Agent X (pbestx)																	
	107	13	3	8	5	15	2	12	14	1	17	4	7	16	6	10	11	9	
gbest		Global best position (gbestx)																	
	61	14	13	12	1	17	11	7	16	15	5	4	8	9	6	10	2	3	
Rndft		Randomly generated position (Randx)																	
	234	3	14	10	9	8	6	7	13	11	17	12	16	1	15	5	2	4	

Figure 5.2: Attributes of a captured instant  $k$  for 17-cities ASTP problem.

It is required to simulate a multi-decision making process in the brain of the agent  $X$  (for simplicity agents are identified by their positions) in order to determine its candidate move  $X^{k+1}$  in accordance with an acquired set of basic behaviors from several resources. This set of basic behaviors consists of imitation, memory retrieval, momentum, and play. Whereas, the best agent in the neighborhood, self attitude (current and successful past) and the attitude acquired randomly constitute the set of mentioned resources. The best individual in the neighborhood  $N$  is characterized by fitness  $Nbest=74$  and a position of  $Nbestx$ . The most successful experience of the given agent  $X$  is specified fitness  $pbest=107$  and position  $pbestx$ . The position that is randomly generated to express play behavior is characterized by fitness  $Rndfit=234$  and position  $Randx$ . The agent with global best ( $gbest$  and  $gbestx$ ) behavior is not directly engaged in the calculations but rather in an indirect way to set limits for fuzzy membership function used to standardize fitness values.

### 5.2.4 Decision alternatives and criteria

For that very instant  $k$  and a given agent  $X$ , the available decision attributes can be arranged as shown in Table 5.1.

**Table 5.1: Decision attributes for agent  $X$  at iteration  $k$**

Alternatives	Fitness	Distance from $X$
Imitation $Nbest$	74	16
Memory $pbest$	107	9
Momentum $X$	235	0
Play $Rndft$	234	15

The values of fitness criteria are obtained by tours' direct substitution into the objective function in Equation (5-1). Diversity criteria are expressed by distances (in genotype space) measured between agent  $X$  and every agent that is engaged in the decision process. Distances are computed using the Levenshtien edit Distance technique [78] and articulate the number of deletions, insertions, or substitutions required for transforming two permutations into each other.

The next step is to estimate the ranked fuzzy values for the decision criteria in order to reach a suitable ranking index by applying fuzzy ordered weighted average aggregation. In this case, the limits required for the membership function are given in Table 5.2.

**Table 5.2: Fuzzification Limits**

Criterion	Limit1	Limit2
Fitness	61	254
Diversity	0	16

Membership functions are constructed according to the ultimate goal: promote solutions that are fit and diverse. For diversity membership, the further the distances, the higher the membership grades. For fitness, the lower the value of objective function (Equation (5-1)) the higher the membership grades.

Accordingly, the ranked fuzzy measures ( $RF$ ) are calculated as in Table 5.3.

**Table 5.3: Ranked fuzzy measures**

	$RF1$	$RF2$
Imitation $Nbest$	1.0000	0.9326
Memory $pbest$	0.7617	0.5625
Momentum $X$	0.0984	0
Play $Rndft$	0.9375	0.1036

### 5.2.5 Decision rule (OWA)

In this step, the OWA is used to reach a single evaluation function. This function measures the performance of each alternative in terms of its criteria. Therefore, agent  $x$  will be able to rank the available alternatives according to the value of that function. The value of this function is described in the text as the ranking index  $RJ$ . To proceed with the calculations, order weights  $W$  have to be specified. For known decision strategies with two criteria, a set of order weights are given in Table 5.4 [81].

Table 5.4: Five different decision strategies [81]

Decision strategy	Order weights
Optimistic	(1,0)
Moderately optimistic	(0.81,0.19)
Neutral	(0.5,0.5)
Moderately pessimistic	(0.13,0.88)
pessimistic	(0,1)

In the simulation, a moderately optimistic decision strategy is chosen (relative risk taking with  $W=(0.81,0.19)$ ).

As a result, the ranking indices values are calculated as:

For imitation alternative:  $RI_i = 0.9872$

For memory retrieval alternative:  $RI_r = 0.7239$

For momentum alternative:  $RI_m = 0.0797$

For play alternative:  $RI_p = 0.7791$

In order to interpret these ranking indices to acquired behaviors by agent  $X$  at the candidate step  $X^{k+1}$ , Equation (5-2) is defined.

$$O_\delta = \text{floor}(n * RI_\delta / \sum_\delta RI) \quad (5-2)$$

Where  $\delta=i, r, m$  or  $p$  and;

$n$ : number of total items in permutation  $\pi$ .

The floor function for a real number  $y$  is denoted by  $\text{floor}(y)$  defined by:

$$\lfloor y \rfloor = \sup\{v \in \mathbb{Z} | v \leq y\};$$

Where  $\mathbb{Z}$  is the set of integer numbers [89].

Equation (5-2) determines the number of items  $O_\delta$  in a position permutation to be transmitted by an agent  $X$  into its candidate position from a particular individual due to an acquired behavior  $\delta$ .



According to Equation (5-2) along with ranking indices values, agent  $X$  acquires six items by imitation ( $O_i = 6$ ), five items by play ( $O_p = 5$ ), and four items by memory retrieval ( $O_r = 4$ ).

Ultimately, the permutation that represents agent  $X$  at iteration  $k+1$  ( $X^{k+1}$ ) will be an objective composition of permutations that represents the available alternatives.

To carry out this composition, permutations are first placed in ascending order according to their ranking order. Second, items are transmitted from the first permutation to permutation  $X$ . Recursively, this process is repeated with the succeeding permutations until the final composed permutation  $X^{k+1}$  is reached as shown in Figure 5.3. The reason the permutations are placed in that order is that one wants the most successful (fit and diverse) alternative to dominate. Note that the transmitted items are chosen randomly.

The entire decision making process is repeated with all agents in all neighborhoods until a set of new positions are obtained. These positions are then employed to get new values for the objective function. For this specific TSP (17 cities), the algorithm was able to reach the optimal tour that cost 39 by checking only a fraction of  $7.0286e-011$  of the total possible tours. The pseudo code of the entire process is shown in Figure 5.4. In addition, the Matlab code and the necessary routines are included in the Appendices.

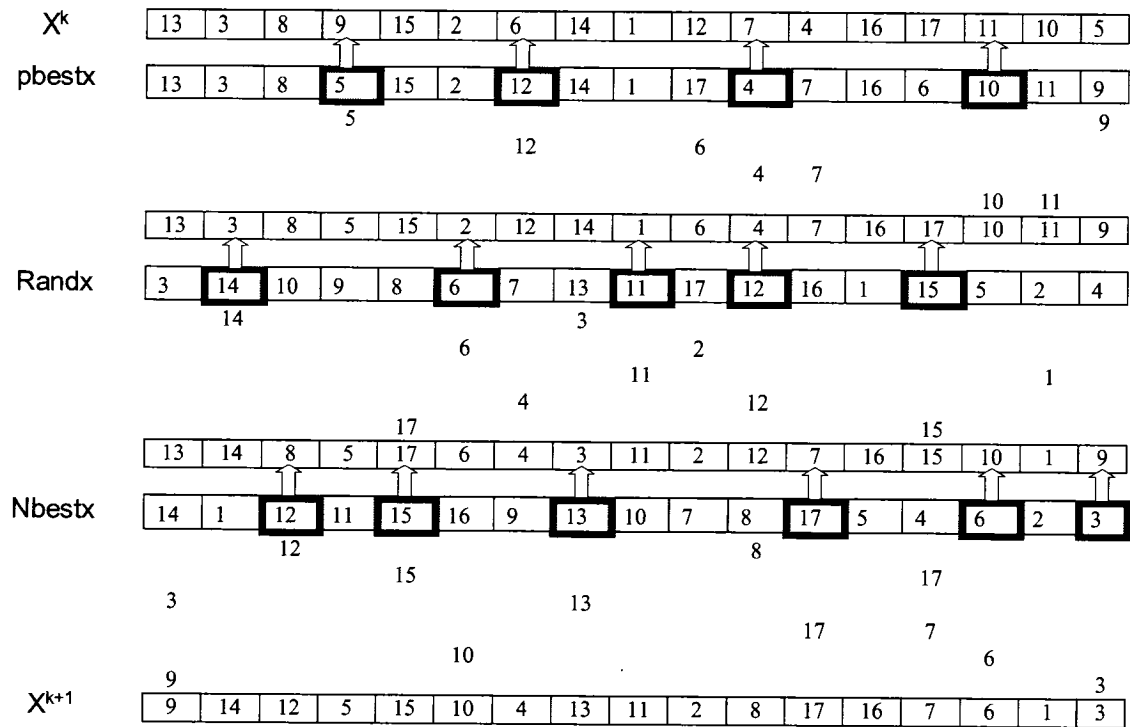


Figure 5.3: Recursive displacement of items among permutations.

```

For Agent = 1: Swarm Size
    Initialize Agent's random position X
    Calculate Agent's Fitness Xft
    Assign Agent's pbest and pbestx
End
Assign global best fitness and corresponding position gbest and gbestx
Divide the swarm into N neighborhoods
For neighborhood = 1: N
    Assign Nbest and Nbestx
End
For k = 1: iterations
    For Agent = 1: Swarm Size
        Create random position for Agent's play behavior Randx
        Calculate Agent's fitness due to random position Rndft
    % Computation of Levenshtein distances LD's
         $LD_r = LD(X, pbestx)$ 
         $LD_i = LD(X, Nbestx)$ 
         $LD_p = LD(X, Randx)$ 
         $LD_m = LD(X, X) = 0$ 
    End
    Set fuzzy limits for the fitness membership function
    FL1 = best fitness (gbest)
    FL2 = worst fitness
    Set fuzzy limits for distance membership function
    DL1 = min (LD's)
    DL2 = max (LD's)
    Construct fuzzy membership functions for fitness and diversity
End
For Agent = 1: Swarm Size
    % Decision criteria standardization
    [A1] = Fuzzified [LDr LDi LDp LDm]
    [A2] = Fuzzified [pbest Nbest Rndft Xft]
    RFr = [max(A11,A21) min(A11,A21)]
    RFi = [max(A12,A22) min(A12,A22)]
    RFp = [max(A13,A23) min(A13,A23)]
    RFm = [max(A14,A24) min(A14,A24)]

```

*The figure continues on the next page*

```

% Calculation of Decision ranking indices RI's
 $RI_r = [RF_r] * [W]^T$ 
 $RI_i = [RF_i] * [W]^T$ 
 $RI_p = [RF_p] * [W]^T$ 
 $RI_m = [RF_m] * [W]^T$ 
Calculate number of items to be acquired by each behavior  $O_\delta$ 
For  $\delta = i, r, m$  and  $p$ 
 $O_\delta = \text{floor} (n * RI_\delta / \sum_\delta RI)$ 
end
Arrange  $X$ ,  $pbestx$ ,  $Nbestx$ ,  $Randx$  in ascending order according to  $RI$ 's
Transmit number of  $O_\delta$  items from the first permutation to  $X$ 
Update  $X$ 
While final permutation in the arrangement is not reached
    Transmit number of  $O_\delta$  items from the arranged permutation to  $X$ 
    Update  $X$ 
End While
New position  $X = \text{updated } X$ 
Calculate Agent's Fitness  $Xft$ 
Update  $pbest$  and  $pbestx$ 
End
Update  $gbest$  and  $gbestx$ 
End Iterations
Solution = ( $gbest$ ,  $gbestx$ )

```

**Figure 5.4: the pseudo code for the main steps of the proposed technique**

### 5.2.6 Larger scale TSP problems

The proposed technique is also tested with larger TSP problems namely; the symmetrical 29 city problem (bays29) and the asymmetrical 43 city problem (p43) [90].

Results from simulations are very promising and are very close to the known optimal solutions [90] as shown in Table 5.5.

**Table 5.5: Simulation results for different TSPs**

No. of cities	optimal	MCDM-PSO	Tested /total tours
ATSP-17	39	39	7.0286e-011
STSP-29	2020	2085	5.6550e-027
ATSP-43	5620	5668	8.2761e-048

50,000 random tours generated over 10 runs for the 43 city problem are inspected. An average fitness of 9646.2 is found. For the same number of evaluations, the corresponding value calculated with the proposed technique (population size 50 and iterations of 1000) was 5684.8.

Also a comparison between the traditional particle swarm optimizer PSO and the proposed technique is held. Figure 5.5 shows the convergence of both MCDM-PSO and the traditional PSO over time (represented by iterations) for the ATSP-43 problem. MCDM-PSO has better performance over the entire search process. The figure also shows that the traditional PSO is trapped at the objective function value of 5870 (before the first 200 iterations) while the MCDM-PSO is able to reach better values. The proposed technique is able to solve the problem to a value of 5655 at iteration 195.

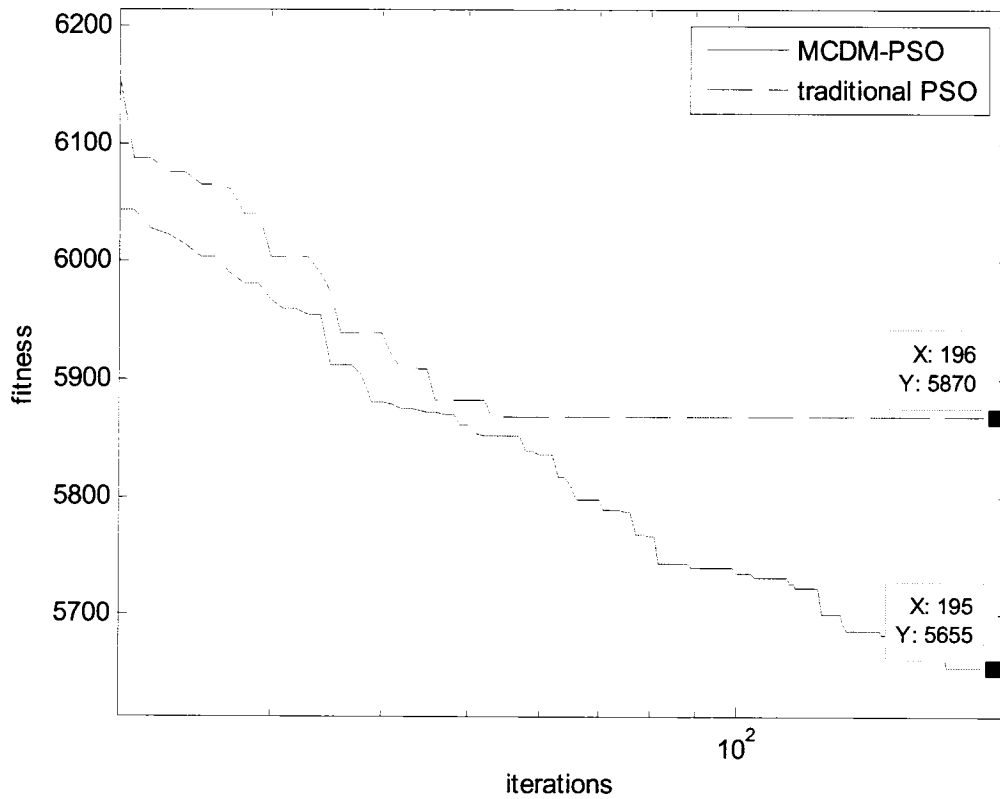


Figure 5.5: Comparison between the traditional PSO and the proposed technique

### 5.3 Quadratic Assignment Problem

Swarm optimizers have been proven to be effective in solving wide classes of combinatorial optimization problems in different disciplines. Surprisingly, the quadratic assignment problem (QAP) appears not to have been fully addressed using swarm optimizers. In this section comprehensive tests on the MCDM-PSO using numerous QAP's problems are conducted. The technique is tested with over a hundred QAP benchmark problems ranging in size from 12 to 256. The chosen benchmark set covers all types of QAP: dense and sparse, randomly generated and real life problems and domains that are highly structured and as well as those that have uniformly distributed local optima. Results are mainly compared with those obtained using a standard genetic algorithm. The performance of the technique is also compared with the performance of the ant colony and traditional PSO algorithms.

The quadratic assignment problem (QAP) has been widely used for testing the performance of heuristic techniques. The problem gains its reputation due to the fact that many practical applications can be stated in terms of the QAP model. Throughout the literature many real-life as well as randomly generated problems have been tackled. Among these applications are hospital facility design (*Nug-type* and *Els-type* problems) [91, 92], computer backboard wiring (*Ste-type* problems) [93], typewriter keyboard design (*Bur-type* problems) [94], testing of self testable sequential circuits (*Esc-type* problems) [95], weighted tree design (*Chr-type* and *Had-type* problems) [96, 97] and facility layout design (*Scr-type* and *Rou-type* problems) [98-100]. Also, another set of randomly generated problems with known optimal solutions can be found in references (*Lipa-type* and *Tai-type* problems) [101-105].

The complexity of the quadratic assignment problem is of the NP-complete type. This means that finding a technique that is able to solve the problem in a polynomial time frame is very unlikely. The time required to reach an optimal solution using an exact technique will grow dramatically with the size of the problem. Complete enumeration of the entire search space is the only way to find the exact solution. The process is very time consuming and unfeasible for even a very moderate problem size. For example solving QAP of size 30 requires evaluating 30! possible assignments. If a trillion assignments are solved each second, it would take around 140 times the age of the universe to optimally solve the problem [106].

Researchers have developed more sophisticated techniques to overcome the computational complexity of the problem. A smarter enumeration process has been used as a basis to develop such techniques. The branch-and-bound method is one of the most successful exact solvers for the problem. The technique was originally introduced in 1960 [107] for linear programming problems. It relies on implicit enumeration by decomposing the root problem into small sub areas constructing a tree-like search procedure. The lower and upper bounds are then used to either prune or branch sub areas. Although the technique of branching and bounding inclusively ensures an exact solution, the time spent

on a sequential solving machine to reach a solution is still unreasonable. For example, it was not until 2001 that *Nug30* has been solved to optimality by branch-and-bound. The experience of solving *Nug30* using the branch-and-bound method is reported in [108]. According to the authors, sequential solving *Nug30* on the most powerful desktop workstation at that time would have taken around 7 years. Therefore, the authors decided to use parallel processing to reduce the computational time. Table 5.6 shows some interesting statistics about this experience.

**Table 5.6: Solving Nug30**

Average number of participating machines	652.7
Maximum number of participating machines	1009
Running wall clock time (days)	6.92
Approximate total cpu time (years)	11

Therefore, there is a need for techniques that could achieve a reasonable trade off between precision and time.

Heuristic techniques can easily find suboptimal solutions for the problem by inspecting only a small fraction of all possible assignments. Therefore, the time required to solve the problem to that sub-optimality can be dramatically reduced.

Simulated annealing (SA) was one of the earliest attempt made to solve QAP in 1980 [109]. Then, a few other SA approaches have been proposed for the QAP [110-112]. The most successful SA approach is attributable to Connolly [113].

Several Tabu search implementations have been also reported in the literature. The first Tabu algorithm for the QAP problem was first introduced by Skorin-Kapov [114] in 1990. Subsequently, some extensions and enhancements to that technique were proposed by the same author [115]. Mechanisms such as short term memory and multiple starts have also



been used to enhance the search diversification of the Tabu search for the QAP [116]. The robust Tabu search introduced by Talillard [117] is the best performing Tabu technique known to deal with the QAP. An anti-stagnation mechanism for the conventional Tabu search has been proposed by Battiti and Tecchiolli [118] to develop the so-called reactive Tabu search.

Ant colony (ACO) based techniques have been employed in tackling the QAP. For example, Maniezzo et al. [119] contributed the first ACO colony implementation for QAP. Several other ACO applications can be found in [120-122]. There are also several techniques based on genetic algorithm and its hybrids [123-125].

### 5.3.1 The assignment problem

In general, the assignment problem involves assigning a set of finite items to a set of locations. A mere sense of items and locations does not apply for every application, but rather depend on the nature of the problem at hand. For example in a job assignment, the set of items are personnel while the locations are the jobs, in a marriage problem items are one gender and locations are the other, and in a hospital layout problem, items are services (departments) and locations are wards in the designated building. To solve the assignment problem one must find the total assignment that maximizes the welfare of all role players. Usually, the objective function is expressed by summation of the cost of all individual assignments.

### 5.3.2 Linear assignment (LAP)

When only one weight matrix (usually called a flow matrix) is involved, the assignment problem is of the type “linear” (LAP). In reference [126] the authors described their LAP model by using the job assignment problem. A set of  $n$  people and set of  $n$  jobs are given. Each person must then be assigned to one and only one job in a way that minimizes the total cost of the assigning process. A cost (or flow) matrix is also given where the matrix element  $c_{ij}$  expresses the cost of assigning a person  $i$  to a job  $j$ . The total cost function can be formulated as:

$$\sum_{i=1}^n c_{i\pi(i)} ; \quad (5-3)$$

Where  $j=\pi(i)$  and  $\pi$  is a bijective mapping of a set of  $n$  integers onto itself. Each possible assignment is expressed by a permutation  $\pi$  so that the orders on this permutation represent the set of people while jobs are represented by elements occupying these orders as shown in Figure 5.6.

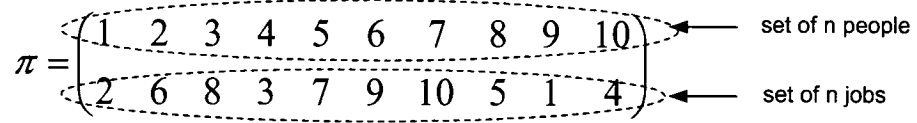


Figure 5.6: Permutation encoding for LAP of  $n=10$ .

It is clear that the individual assignment of person  $i$  to job  $j$  is made independently and does not influence any other assignment. Consequently, the cost of that assignment depends only on the affinity between this particular pair, a feature that gives the problem its linearity.

### 5.3.3 The quadratic assignment problem (QAP)

To facilitate a comparison between LAP and QAP, the same job assignment model is also used here but with a slight change: the jobs are replaced by offices. The problem now is assigning  $n$  people to  $n$  offices. In this case, a flow matrix  $C$  is given, where the element  $c_{ij}$  represents the volume of flow of activities between person  $i$  and person  $j$ . A matrix  $D$  is also given where the element  $d_{kl}$  represents the distance between office  $k$  and office  $l$ . The cost of an individual assignment is now calculated as the flow of activities between person  $i$  and person  $j$  times the distance between the office  $k$  (assigned for person  $i$ ) and office  $l$  (assigned to person  $j$ ). Then, the objective function to be minimized is the double summation of the products of flows and distances over all  $n$ .

Mathematically, the objective function can be expressed by:

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} d_{\pi(i)\pi(j)} ; \quad (5-4)$$

Where  $k=\pi(i)$ ,  $l=\pi(j)$  and  $\pi$  is a bijective mapping of a set of  $n$  integers onto itself. A potential assignment is expressed by a permutation  $\pi$  where the orders in this permutation represent the set of  $n$  offices and elements occupying these orders represent the set of  $n$  people. This formulation is known as the K-B statement which was originally suggested by T. C. Koopmans and M. J. Berkmann [127] in 1957 to solve the facilities allocation problem.

Another formulation of QAP was also introduced by the same authors by assuming a binary variable  $x_{ij}$  that takes value of 1 if facility  $i$  is allocated to position  $j$  and zero otherwise. Then the problem can be stated as [128]:

$$\min \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n c_{ij} d_{kl} x_{ik} x_{jl}, \quad (5-5)$$

Subject to:

$$\sum_{i=1}^n x_{ij} = 1, \quad \forall j$$

$$\sum_{j=1}^n x_{ij} = 1, \quad \forall i$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1, \dots, n.$$

Unlike LAP, in the QAP an individual assignment nonlinearly influences other assignments especially with people who have nonzero entries in the flow matrix with this particular person. The quadratic nature of the problem is a result of using an objective function that has a term of the second degree.

### 5.3.4 QAP benchmarks

It is known that the performance of a heuristic search technique is highly influenced by the nature of the problem studied. Among QAP problems, discrepancies are very high due to the wide range of applications that QAP can cover. These involve the topography of the fitness landscape and the sparsity of both flow and distance matrices. In the

literature, the QAP problems are named after the authors who first introduced it. Usually the first three (some times four) alphabetic letters in the problem name denote the author name and the immediately following number refers to the size of the problems (for example, the *Nug30* is a QAP of size 30 due to C. Nugent [91]). For some problems, different flow and cost matrices structures for the same problem can be found. In this case, the problem name is distinguished by small letters at the end of the name (for example, *Tai35a* and *Tai35b*). The distribution of local optima over the search space is an important factor when determining the hardness of QAP at hand. Researchers have agreed upon three main categories of QAPs regardless of their application. According to [128] and [129] these three categories are:

- I. Unstructured problems: In this case local minima are uniformly distributed over the entire space. Random problems that were originally generated to test the robustness of heuristic algorithms can be classified under this category. This type is considered the hardest to solve to optimality [128]. Examples of this category are *Tai25a*, *Tai100a* and *Rou20*.
- II. Strongly structured problems: This type includes problems that have concentrated local optima in their search space. Usually, real-life problems and those generated to resemble real-life problems are classified under this class. Examples of these problems are *Els19*, *Nug30* and *Tai35b*.
- III. Structured problems: In this category local minima are distributed over spread clusters in the search space. *Tai100b*, *Wil100* and *Tho150* are examples of this category.

Figure 5.7 is a copy from reference [129] and shows the main categories of QAP landscapes.

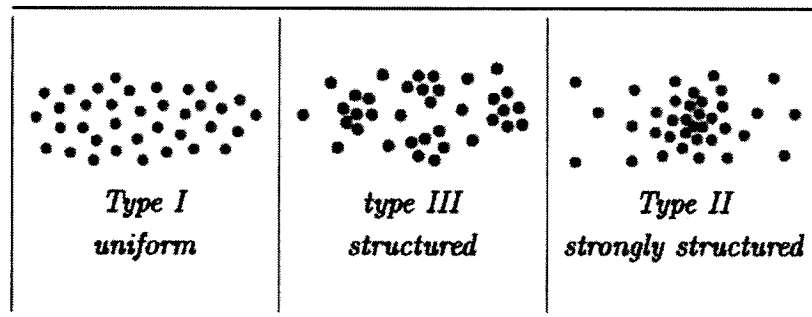


Figure 5.7: Classification of QAP benchmarks [129].

### 5.3.5 Testing and results

The algorithm is tested with 131 QAP problems that vary in size and also in difficulty level according to the distribution of their local optima in the search landscape. Data for all the tested problems are available at the online library of the quadratic assignment problem QAPLIB [94]. For the simulations, the number of evaluations is kept fixed by controlling both the population size and iteration number. For this purpose, evaluations are considered to be the population size multiplied by the maximum number of iterations. Because the set of tested problems are relatively large and sizes are widely varied, the number of evaluations is chosen to be 30,000. In easy and/or small size problems, a population size of 15 and maximum iterations of 2000 are used. For hard and/or large size problems, population size and maximum number of iterations are set to be 30 and 1000 respectively. The algorithm is coded in Matlab 7 and the results are obtained by running the algorithm on a Pentium M 725 personal computer with the speed of 1.6 GHZ. Ten runs for each problem are carried out and then descriptive statistics are reported. The parameters of the standard genetic algorithm SGA [130] used for comparison are as follows:

Population size=100

Maximum generation=500

Fitness scaling factor=3

Crossover probability=0.8

Mutation probability=0.05

In Tables 5.7 to 5.18, the best known solution of each tested problem is stated as BKS. The heading AVG represents the average value found for the objective function over ten runs. AVG Gap heading refers to the percentage of error between the best known and the average solutions. The Gap only heading represents the difference between the BKS and the best solution reached by both algorithms.

For *Nug* type instances with sizes less than or equal to 22, Table 5.7 shows the average performance of SGA with gap between 3-5% while the average performance of MCDM-PSO always has a gap of around 1%. For the larger size *Nug* type problems, the average gap for solutions found by using SGA are greater than 5% but for MCDM-PSO this value is under 3%. It is also noticeable that MCDM-PSO outperforms the SGA when comparing the best results obtained by both algorithms even for the hardest *Nug* problems (*Nug30*). In general, the best values obtained by the algorithm are always lower than 0.75 while in the case of SGA the gap reaches values above 4%.

In Table 5.8, the results for *Bur* type problems are reported. *Bur* type problems are relatively easy and the accuracy of solutions found by both SGA and MCDM-PSO exceeds 99 % in most cases. However, the average and best performances of MCDM-PSO are significantly better than SGA even in the narrow range of that 1% error gap.

For *Chr* type problems (Table 5.9), the average performance of SGA is very poor and in some cases the average gap reaches values over 30%. Although the average results obtained by MCDM-PSO for 7 out of the 14 problems are not quite satisfactory, they are still significantly better than those of SGA. For those seven problems the average gap can be reduced by almost 50% using MCDM-PSO. There is also a noticeable improvement in the best optima obtained using MCDM-PSO. In four cases optimal solutions are found and for the rest of the problems the error is reduced by half by employing the technique.

For *Esc* and *Els* type problems shown in Table 5.10, although the best performance of SGA is relatively good, the average performance is very poor. The SGA is able to find optimal solutions for most cases. However, the stability of the SGA is very low as the average gap for some problems hits values over 80%. For example, while the SGA is able to locate the optimal value for the problem *Esc32f*, the average gap is 85.32%, which reflects low consistency in finding a reasonable solution in general.

On the other hand, the average performance of MCDM-PSO is very good and significantly better than SGA. The algorithm shows high consistency with almost all problems of this set (except for *Esc128*). The algorithm is able to find optimal solutions for the ten problems at every run. The best solution found by MCDM-PSO is always equal to the best known solution for all problems except for *Esc32a*, *Esc32h* and *Esc128*. The problem with the size 128 (*Esc128*) is quite large and relatively hard to solve. This is very clear with SGA as the average and best gaps are 63.64% and 52.24% respectively. However, for the same problem with MCDM-PSO the solution has significantly improved as the average and best gaps are reduced to 21.38% and 13.51% respectively.

Simulation results for *Had*, *Kra*, *Scr*, *Rou*, *Tho*, *Wil*, and *Ste* type problems are reported in Tables 5.11 to 5.13. Except for *Had* type problems, the average and best performance for SGA are very poor compared to MCDM-PSO. For SGA the average gap reported is between 4-35% and gap of 1.88-28%. The average gap in the case of MCDM-PSO for this set is always around 4%. However, the algorithm is able to reach solutions with accuracy over 96% during ten runs for all problems including hard and large ones.

Randomly generated problems *Sko*, *Tai*, and *Lipa* types have large sizes and are mainly generated to test the performance of heuristic techniques. Results for these problems are reported in Tables 5.14 to 5.18. For *Sko* type instances the average gap for SGA is always around 8% while for MCDM-PSO the value of the average gap is around 3%. For the best performance comparison, the gap reached by SGA is around 7% but for MCDM-PSO this value is only 2%.

For *Tai-a* and *Tai-c* type, an improvement is achieved using the proposed MCDM-PSO for both average and best performances. The average gap obtained using SGA ranges from 4% to 10% while for MCDM-PSO this value is 1% to 7%. For *Tai-b* type problems (Table 5.16), while both average and best performances for SGA are poor, MCDM-PSO has maintained the same level of improvement as with *Tai-a* and *Tai-c* types.

Table 5.17 shows simulation results for *Lipa-a* type. This type is relatively easy and the performance of SGA is satisfactory. However, MCDM-PSO still shows superiority over SGA in terms of average and best gaps obtained in this case. Although the *Lip-b* type is much harder and performances of both algorithms are not acceptable, MCDM-PSO is able to reach better solutions than those obtained by SGA (Table 5.18).

**Table 5.7: Results for Nug-type problems**

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Nug12	578	603.8	4.27%	582	0.69%	586.6	1.47%	578	0.00%
Nug14	1014	1057.4	4.10%	1018	0.39%	1030	1.55%	1014	0.00%
Nug15	1150	1190.6	3.41%	1160	0.86%	1157.2	0.62%	1150	0.00%
Nug16a	1610	1679	4.11%	1610	0	1623.4	0.83%	1612	0.12%
Nug16b	1240	1292	4.02%	1268	2.21%	1254	1.12%	1240	0.00%
Nug17	1732	1803.8	3.98%	1744	0.69%	1756.8	1.41%	1732	0.00%
Nug18	1930	2025	4.69%	1982	2.62%	1968.8	1.97%	1944	0.72%
Nug20	2570	2695.2	4.65%	2648	2.95%	2614	1.68%	2570	0.00%
Nug21	2438	2566.2	5.00%	2478	1.61%	2478.2	1.62%	2450	0.49%
Nug22	3596	3756.2	4.26%	3664	1.86%	3648.8	1.45%	3596	0.00%
Nug24	3488	3719.8	6.23%	3648	4.39%	3589.6	2.83%	3510	0.63%
Nug25	3744	3952.4	5.27%	3864	3.11%	3797	1.40%	3772	0.74%
Nug30	6124	6549	6.49%	6346	3.50%	6316	3.04%	6168	0.71%

**Table 5.8: Results for Bur-type problems**

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Bur26a	5426670	5457625	0.57%	5441008	0.26%	5436784.2	0.19%	5432875	0.11%
Bur26b	3817852	3840110	0.58%	3826765	0.23%	3826543.3	0.23%	3824643	0.18%
Bur26c	5426795	5453160	0.48%	5436828	0.18%	5432053.1	0.10%	5427227	0.01%
Bur26d	3821225	3843722	0.59%	3824800	0.09%	3826175.6	0.13%	3821427	0.01%
Bur26e	5386879	5431801	0.83%	5400658	0.26%	5390803.8	0.07%	5387239	0.01%
Bur26f	3782044	3805317	0.61%	3784041	0.05%	3784339.5	0.06%	3782696	0.02%
Bur26g	10117172	10203375	0.84%	10146856	0.29%	10124142	0.07%	10118542	0.01%
Bur26h	7098658	7149410	0.71%	7103022	0.06%	7114724	0.23%	7099216	0.01%



Table 5.9: Results for Chr-type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Chr12a	9552	12931	26.13%	10214	6.48%	10229.4	6.62%	9552	0.00%
Chr12b	9742	11755.6	17.13%	9742	0	10648.8	8.52%	9742	0.00%
Chr12c	11156	13675.2	18.42%	11156	0	11804	5.49%	11156	0.00%
Chr15a	9896	14167.2	30.15%	11634	14.94%	10940.6	9.55%	9896	0.00%
Chr15b	7990	13396.6	40.36%	10004	20.13%	9581.8	16.61%	8640	7.52%
Chr15c	9504	14118	32.68%	11310	15.97%	11792.2	19.40%	10678	10.99%
Chr18a	11098	19065.4	41.79%	14418	23.03%	14832.8	25.18%	12680	12.48%
Chr18b	1534	1778.8	13.76%	1570	2.29%	1627	5.72%	1556	1.41%
Chr20a	2192	3340.8	34.39%	2822	22.32%	2699.8	18.81%	2554	14.17%
Chr20b	2298	3397.6	32.36%	2954	22.21%	2765.6	16.91%	2678	14.19%
Chr20c	14142	39151.2	63.88%	19696	28.20%	22110.4	36.04%	17546	19.40%
Chr22a	6156	7240.8	14.98%	6934	11.22%	6598.2	6.70%	6364	3.27%
Chr22b	6194	7254.6	14.62%	6840	9.44%	6499.8	4.70%	6292	1.56%
Chr25a	3796	6492	41.53%	5734	33.80%	5126.4	25.95%	4582	17.15%

Table 5.10: Results for Esc-type and Els-types problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Esc16a	68	70	2.86%	68	0	68	0.00%	68	0.00%
Esc16b	292	292	0	292	0	292	0.00%	292	0.00%
Esc16c	160	162	1.23%	160	0	160	0.00%	160	0.00%
Esc16d	16	16.6	3.61%	16	0	16	0.00%	16	0.00%
Esc16e	28	30.8	9.09%	28	0	28	0.00%	28	0.00%
Esc16g	26	28.4	8.45%	26	0	26	0.00%	26	0.00%
Esc16h	996	996	0	996	0	996	0.00%	996	0.00%
Esc16i	14	14	0	14	0	14	0.00%	14	0.00%
Esc16j	8	9.8	18.37%	8	0	8	0.00%	8	0.00%
Esc32a	130	178.4	27.13%	166	21.69%	143.8	9.60%	136	4.41%
Esc32b	168	231.2	27.34%	212	20.75%	187.6	10.45%	168	0.00%
Esc32c	642	645	0.47%	642	0	642	0.00%	642	0.00%
Esc32d	200	214	6.54%	200	0	200.8	0.40%	200	0.00%
Esc32e	2	13.8	85.51%	2	0	2	0.00%	2	0.00%
Esc32f	2	13.8	85.51%	2	0	2	0.00%	2	0.00%
Esc32g	6	8.2	26.83%	6	0	6	0.00%	6	0.00%
Esc32h	438	462.4	5.28%	446	1.79%	442	0.90%	440	0.45%
Esc64a	116	138.6	16.31%	120	3.33%	118.6	2.19%	116	0.00%
Esc128	64	176	63.64%	134	52.24%	81.4	21.38%	74	13.51%
Els19	17212548	20337423	15.37%	17435470	1.28%	17522349	1.77%	17212548	0.00%

Table 5.11: Results for Had-type and Kra-type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Had12	1652	1668	0.96%	1654	0.12%	1658.6	0.40%	1652	0.00%
Had14	2724	2737.8	0.50%	2724	0	2726.4	0.09%	2724	0.00%
Had16	3720	3740.2	0.54%	3720	0	3721.2	0.03%	3720	0.00%
Had18	5358	5416.2	1.07%	5374	0.30%	5383.2	0.47%	5358	0.00%
Had20	6922	6976.6	0.78%	6922	0	6945.2	0.33%	6922	0.00%
Kra30a	88900	98628	9.86%	95780	7.18%	92837	4.24%	91860	3.22%
Kra30b	91420	101470	9.90%	99540	8.16%	93652	2.38%	92170	0.81%

Table 5.12: Results for Scr-type and Rou-type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Scr12	31410	33040	4.93%	32236	2.56%	31857.2	1.40%	31410	0.00%
Scr15	51140	56394	9.32%	54874	6.80%	53089.4	3.67%	51140	0.00%
Scr20	110030	126192	12.81%	118362	7.04%	114319.4	3.75%	110994	0.87%
Rou12	235528	245944.8	4.24%	240038	1.88%	240143.6	1.92%	235528	0.00%
Rou15	354210	382082.4	7.29%	372560	4.93%	366093	3.25%	359748	1.54%
Rou20	725522	766679	5.37%	751964	3.52%	744858.4	2.60%	738232	1.72%

Table 5.13: Results for Tho-type, Wil-type and Ste-type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Tho30	149936	160725.4	6.71%	157034	4.52%	154728	3.10%	154170	2.75%
Tho40	240516	266806	9.85%	262936	8.53%	255916	6.02%	255290	5.79%
Tho150	8133864	9082809	10.45%	9049974	10.12%	8532834	4.68%	8520968	4.54%
Wil50	48816	50993	4.27%	50456	3.25%	50034.8	2.44%	49946	2.26%
Wil100	273038	286112.4	4.57%	284734	4.11%	282245.9	3.26%	277690	1.68%
Ste36a	9526	12113.2	21.36%	11676	18.41%	10542.9	9.65%	10043	5.15%
Ste36b	15852	24547.8	35.42%	21746	27.10%	16723.9	5.21%	16422	3.47%
Ste36c	8239110	10065947	18.15%	9499452	13.27%	8832323.6	6.72%	8630764	4.54%

Table 5.14: Results for Sko-type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Sko42	15812	17105.6	7.56%	16936	6.64%	16412.8	3.66%	16322	3.12%
Sko49	23386	25255.4	7.40%	24990	6.42%	23865.8	2.01%	23782	1.67%
Sko56	34458	37309.8	7.64%	36858	6.51%	35456.2	2.82%	35018	1.60%
Sko64	48498	52669.2	7.92%	52252	7.18%	50470.4	3.91%	50418	3.81%
Sko72	66256	71864.6	7.80%	71202	6.95%	68156.2	2.79%	68082	2.68%
Sko81	90998	98693.6	7.80%	98326	7.45%	93575.6	2.75%	93434	2.61%
Sko90	115534	125670	8.07%	125026	7.59%	119033.4	2.94%	118858	2.80%
Sko100a	152002	164847.8	7.79%	163894	7.26%	156678.8	2.98%	156560	2.91%
Sko100b	153890	166754	7.71%	166344	7.49%	158286.2	2.78%	158084	2.65%
Sko100c	147862	161477.2	8.43%	159682	7.40%	152013	2.73%	151778	2.58%
Sko100d	149576	162531.6	7.97%	161292	7.26%	153993.2	2.87%	153816	2.76%
Sko100e	149150	162335.4	8.12%	161262	7.51%	153467	2.81%	153232	2.66%
Sko100f	149036	161135.8	7.51%	159770	6.72%	153098.8	2.65%	152968	2.57%

Table 5.15: Results for Tai-a and Tai-c type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Tai12a	224416	243950.2	8.01%	229092	2.04%	236794.4	5.23%	229092	2.04%
Tai15a	388214	405865	4.35%	399716	2.88%	397486	2.33%	391586	0.86%
Tai17a	491812	520010.6	5.42%	511354	3.82%	505586.3	2.72%	496906	1.03%
Tai20a	703482	751416	6.38%	734536	4.23%	734276.5	4.19%	716954	1.88%
Tai25a	1167256	1253397	6.87%	1233592	5.38%	1210320.2	3.56%	1202804	2.96%
Tai30a	1818146	1933424	5.96%	1907086	4.66%	1881280	3.36%	1859570	2.23%
Tai35a	2422002	2602995	6.95%	2581634	6.18%	2562068	5.47%	2528876	4.23%
Tai40a	3139370	3378429	7.08%	3344316	6.13%	3256898	3.61%	3232998	2.90%
Tai50a	4941410	5357806	7.77%	5290886	6.61%	5252999.9	5.93%	5183240	4.67%
Tai60a	7208572	7832143	7.96%	7759426	7.10%	7642445.7	5.68%	7590492	5.03%
Tai80a	13557864	14633429	7.35%	14572116	6.96%	14531588	6.70%	14146664	4.16%
Tai100a	21125314	22787707	7.30%	22714742	7.00%	22268020	5.13%	21907198	3.57%
Tai64c	1855928	2049224.8	9.43%	1936770	4.17%	1891672	1.89%	1863678	0.42%
Tai256c	44759294	47384658	5.54%	46530542	3.81%	46169617	3.05%	45934596	2.56%

Table 5.16: Results for Tai-b type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Tai12b	39464925	41082684	3.94%	39464925	0	39912226	1.12%	39464925	0.00%
Tai15b	51765268	52013565	0.48%	51932236	0.32%	51916044	0.29%	51838283	0.14%
Tai20b	122455319	135380782	9.55%	124094680	1.32%	125718242	2.60%	122455319	0.00%
Tai25b	344355646	394382476	12.68%	354169763	2.77%	363005468	5.14%	350828954	1.85%
Tai30b	637117113	734364132	13.24%	663712392	4.01%	669087796	4.78%	638683376	0.25%
Tai35b	283315445	313592869	9.66%	294359840	3.75%	287713908	1.53%	286691656	1.18%
Tai40b	637250948	725153545	12.12%	686649955	7.19%	685075188	6.98%	642497530	0.82%
Tai50b	458821517	533230371	13.95%	506651344	9.44%	491245335	6.60%	490508067	6.46%
Tai60b	608215054	712435922	14.63%	675902852	10.01%	647761636	6.11%	646460819	5.92%
Tai80b	818415043	958168279	14.59%	932742966	12.26%	866025842	5.50%	863647828	5.24%
Tai100b	1.186E+09	1.403E+09	15.45%	1.377E+09	13.88%	1.312E+09	9.62%	1.241E+09	4.41%
Tai150b	498896643	576996377	13.54%	571111565	12.64%	531089719	6.06%	530444728	5.95%

Table 5.17: Results for Lipa-a type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Lipa20a	3683	3793.6	2.92%	3777	2.49%	3759	2.02%	3683	0.00%
Lipa30a	13178	13475.2	2.21%	13448	2.01%	13465	2.13%	13427	1.85%
Lipa40a	31538	32119.7	1.81%	32075	1.67%	32016	1.49%	31988	1.41%
Lipa50a	62093	63168.2	1.70%	63114	1.62%	63064	1.54%	63030	1.49%
Lipa60a	107218	108862.4	1.51%	108831	1.48%	108641.8	1.31%	108540	1.22%
Lipa70a	169755	172096.1	1.36%	171972	1.29%	171566.1	1.06%	171352	0.93%
Lipa80a	253195	256313.3	1.22%	256217	1.18%	255744.6	1.00%	255520	0.91%
Lipa90a	360630	364924.2	1.18%	364748	1.13%	363934.6	0.91%	363594	0.82%

Table 5.18: Results for Lipa-b type problems

		SGA				MCDM-PSO			
Instance	BKS	AVG	AVG Gap	Best	Gap	AVG	AVG Gap	Best	Gap
Lipa20b	27076	31462.8	13.94%	30648	11.65%	28895	6.30%	27076	0.00%
Lipa30b	151426	178665.2	15.25%	176571	14.24%	172469	12.20%	167524	9.61%
Lipa40b	476581	577060.2	17.41%	572405	16.74%	571770.5	16.65%	564443	15.57%
Lipa50b	1210244	1466831	17.49%	1457456	16.96%	1445058	16.25%	1432712	15.53%
Lipa60b	2520135	3098152	18.66%	3086459	18.35%	3052813.2	17.45%	3017804	16.49%
Lipa70b	4603200	5701003	19.26%	5679828	18.96%	5592340.9	17.69%	5573648	17.41%
Lipa80b	7763962	9707771	20.02%	9669701	19.71%	9494589	18.23%	9442384	17.78%
Lipa90b	12490441	15643342	20.15%	15592856	19.90%	15378368	18.78%	15280839	18.26%

In general, the MCDM-PSO is able to find sub optimal solutions for 115 out of 131 problems with a gap around 5%. Out of that set, 40 problems are solved to optimum and 80 problems are solved with accuracy around 99%. To have better visualization of the difference in error gaps in the cases of MCDM-PSO and SGA, the distributions of the error gaps for both average and best performances are plotted in Figures 5.8 and 5.9.

For instance, normality regarding the distribution of error gaps is assumed just to show the mean value of the error and its dispersion in each case. The smaller the mean value the better the performance and the larger the dispersion the lower the consistency of algorithm. In the analysis, the term consistency is defined as the ability of a particular algorithm to find high accuracy solutions over larger portion of studied problems.

For average performance, the MCDM-PSO has a mean value of 6.56 and a standard deviation of 8.16, whereas for SGA these values are 11.89 and 14.61 respectively (Figure 5.8). For best performance, while MCDM-PSO has error gaps with a mean of 2.95 and a standard deviation of 4.6, the SGA algorithm has values of 4.79 and 6.22 respectively (Figure 5.9). This leads to the conclusion that the MCDM-PSO has better average and the best performances, as well as higher consistency than the SGA.

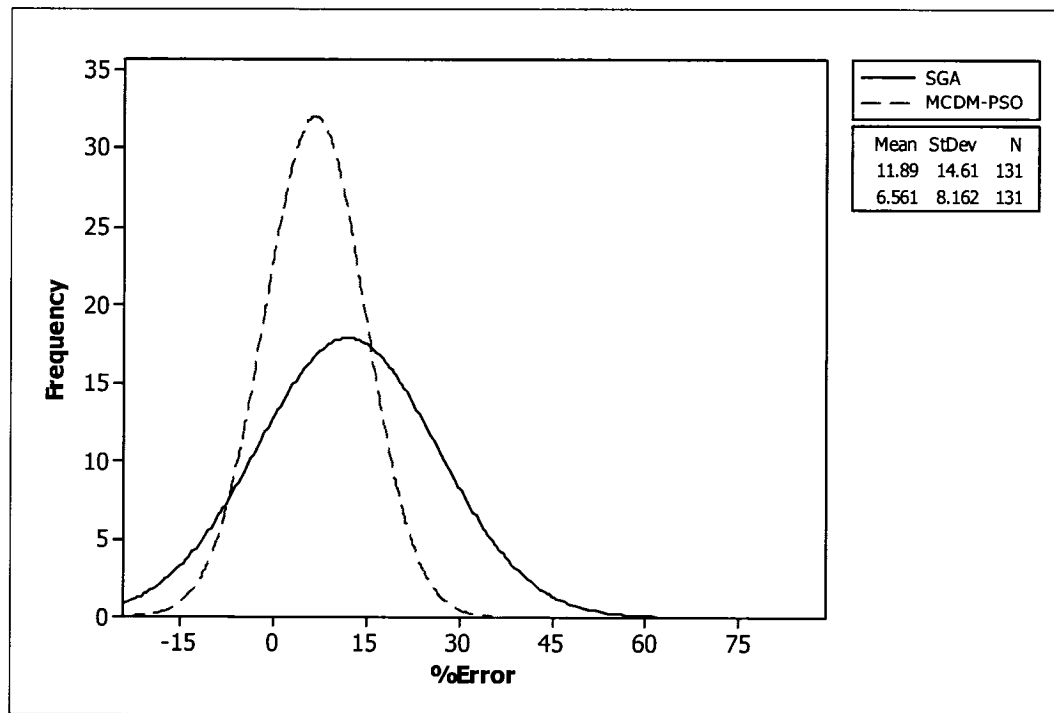


Figure 5.8: Differences between gaps assuming normality (average case)

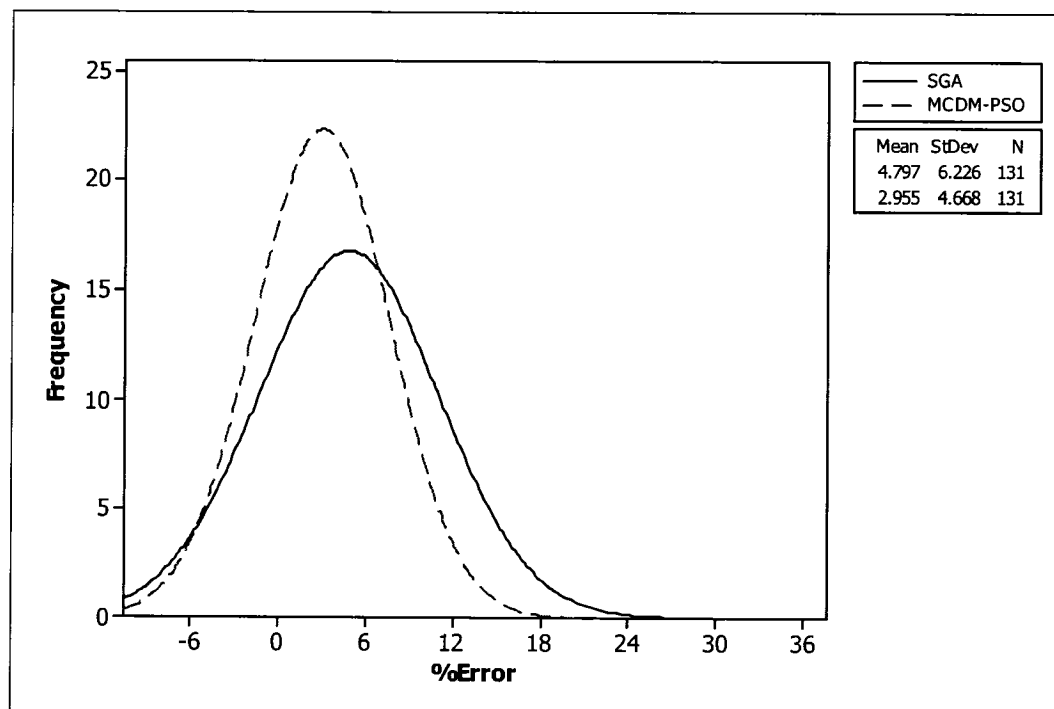


Figure 5.9: Differences between gaps assuming normality (best case).

## 5.4 Significance of error differences

In this section, error gaps obtained by the MCDM-PSO and those obtained by the SGA are statistically tested to prove that the differences are not due to chance. In the previous section, normality regarding the distribution of error gaps is assumed to facilitate a visual comparison. To go further and test the significance of the error difference, one cannot simply assume normality; a statistical test is required to prove normality. This step is very important in determining the type of test that can be performed on the error samples. For samples that come from normally distributed populations, a parametric test should be implemented. For samples that are not normally distributed, nonparametric test types are the appropriate to carry out. In this case, the Anderson-Darling test is performed to test whether the error gap samples are drawn from normally distributed populations or not.

### 5.4.1 Anderson-Darling test of normality

In this test, the null hypothesis is evaluated as  $H_0$ : samples follow a normal distribution versus the alternative hypothesis  $H_1$ : samples do not follow a normal distribution. The test is carried out using Minitab 14.2 with a chosen level of significance  $\alpha$  of 0.1. If the samples are perfectly normal, then the sample on the probability plot can be reasonably fitted by a straight line. The outcomes of the test show that none of the error gaps samples follows a normal distribution due to the lower values of  $p$  obtained in each case. From Figure 5.10, it is clear that the  $p$ -value in each case is lower than the chosen level of significance. There is therefore not enough evidence believe that samples follow a normal distribution and the null hypothesis can be rejected at 0.1-level of significance.

Accordingly, a nonparametric test should be chosen to test the significance of the difference between the error gaps obtained by MCDM-PSO and SGA.

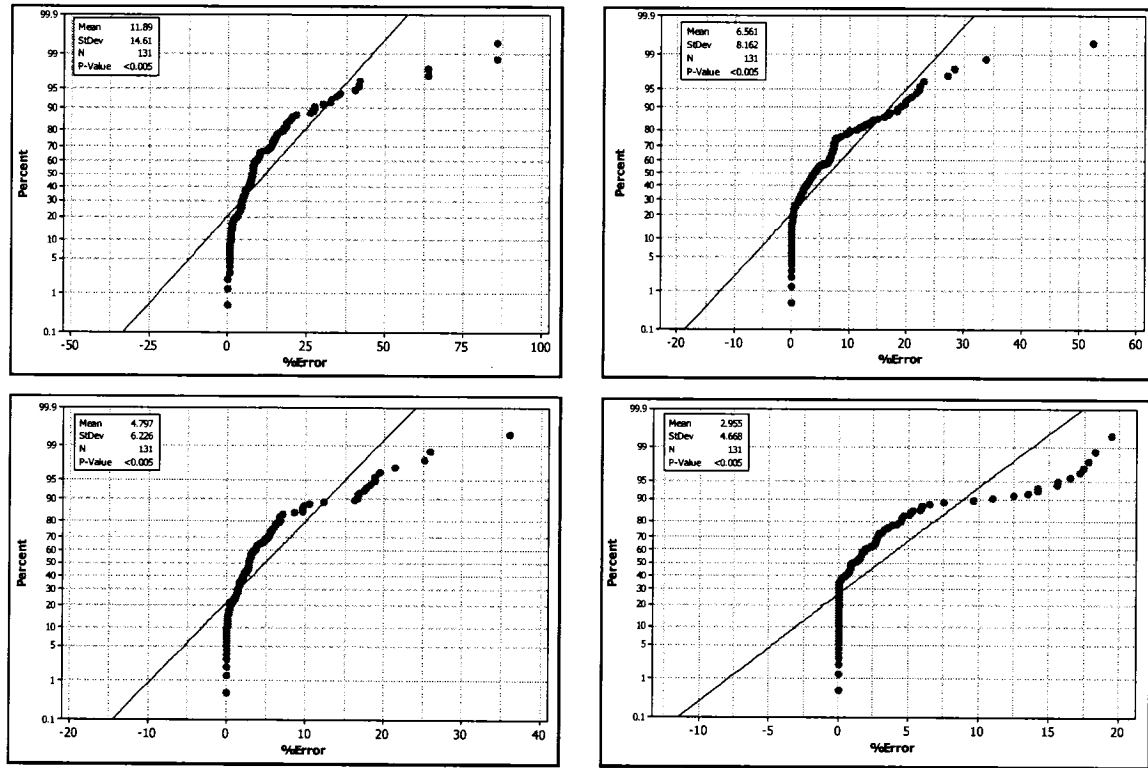


Figure 5.10: Anderson-Darling test results

### 5.4.2 Mann-Whitney test

Mann-Whitney [131] is a nonparametric type of test that should be implemented when the assumption of normality for certain distribution is violated. This test was originally independently developed by Mann and Whitney [131] and Wilcoxon [132] to test the significance of the difference between two random variables using their medians. According to Mann and Whitney [131], if  $x$  and  $y$  are two random variables with  $n$  and  $m$  observations respectively; the variable  $y$  can be declared stochastically larger than  $x$  if:

$$cdf_y(a) < cdf_x(a), \quad \forall a;$$



where  $cdf$  is the cumulative distribution function that can be described as shown in Figure 5.11.

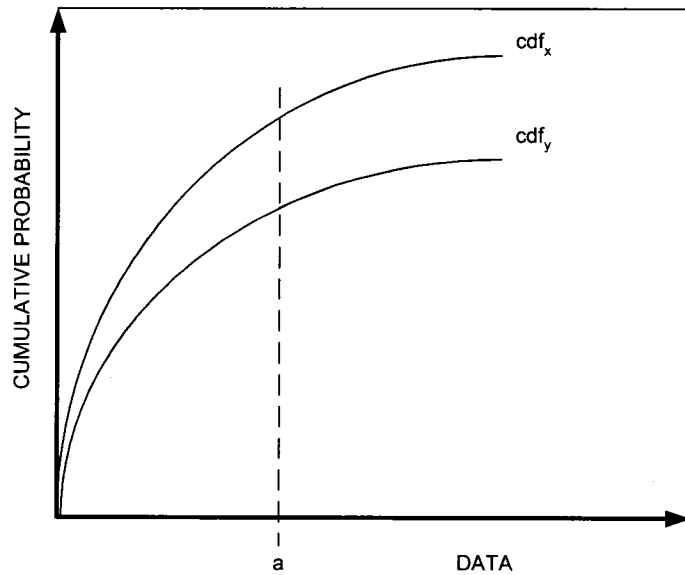


Figure 5.11: Cumulative distribution function

### 5.4.3 U-statistic

The Mann-Whitney test is based on pairwise comparison of two random variables. Observations from both variables are arranged in one increasing order then every value is given a rank depending on its order. The sum of ranks is then used to test whether or not the two samples are drawn from the same population. In order to achieve this, a statistic called  $U$  was proposed by Mann and Whitney [131]. This statistic reflects how many times each random variable precedes the other one in the order. For example,  $U_x$  counts the number of times  $y$  proceeds  $x$  (or  $x > y$ ). Under the null hypothesis  $U$  has known distribution that is tabulated in many statistics books for number of samples less than or equal to 20. For larger sample numbers, the normal distribution approximation described in the next subsection can be used.

Generally, one wishes to test the null hypothesis  $H_0$ :

$$cdf_y = cdf_x;$$

against the alternative hypothesis  $H_1$ :

$$cdf_y < cdf_x.$$

Mann and Whitney [131] defined the decision on the test in case of  $P(U_x \leq U_y) = \alpha$  as accepting the alternative hypothesis that the samples in  $x$  are significantly smaller than the samples in  $y$  and the level of significance is  $\alpha$  if  $U_x \leq U_y$ .

#### 5.4.4 Performing the test

In order to perform the test, the next few steps can be followed to calculate the  $U$ -statistic and thereby make the decision.

1. Arrange observations from both variables in one increasing order.
2. Indicate which observation is related to which variable.
3. Rank every observation according to its position. Ties are dealt with by averaging the ranks over common values under both variables.
4. Calculate the sum of the ranks for each variable as  $R_n$  and  $R_m$ .
5. Calculate  $U$  for both samples as:

$$U_x = nm + \frac{m(m+1)}{2} - R_m \quad (5-6)$$

$$U_y = nm + \frac{n(n+1)}{2} - R_n \quad (5-7)$$

6. Calculate  $U$ -statistic as the smallest value between  $U_x$  and  $U_y$  ( $\min\{U_x, U_y\}$ ).
7. Use the Mann-Whitney statistical test table to locate the critical value of  $U$ . Reject the null hypothesis if  $U$ -statistic is equal to or smaller than the critical value of  $U$  at the specified level of significance.

The previous procedure is applied only in cases when the number of samples is less than or equal to 20. However in the case of larger sample sizes, normal approximation for  $U$ -statistic can be used as follows:

$z = \frac{U - \mu_U}{\sigma_U}$ ; with mean value of U as :

$\mu_U = \frac{nm}{2}$  , and standard deviation as:

$$\sigma_U = \sqrt{\frac{nm(N+1)}{12}} \text{ and } N = n + m$$

In the case of normal approximation, ties adjustment also has to be made to the standard deviation of U. The equation for the standard deviation becomes [133]:

$$\sigma_U = \sqrt{\frac{nm}{N(N-1)} \times \left[ \frac{N^3 - N}{12} - \sum_{j=1}^g \frac{t_j^3 - t_j}{12} \right]} \quad (5-8)$$

where;

$g$  is the number of groups of ties

$t_j$  is the number of tied ranks in group  $j$ .

To apply the test to the results, data are divided into two cases:

Case 1: error (gaps) for average results

Case 2: error (gaps) for best results.

In each case there are two groups of independent samples (SGA and MCDM-PSO) with equal number of samples ( $n=m=131$ ).

The possibility that the variables are normally distributed has been rejected based on the results of Anderson-Darling test explained earlier. Figures 5.12 and 5.13 show nonparametric distribution fitting for error samples in each case.

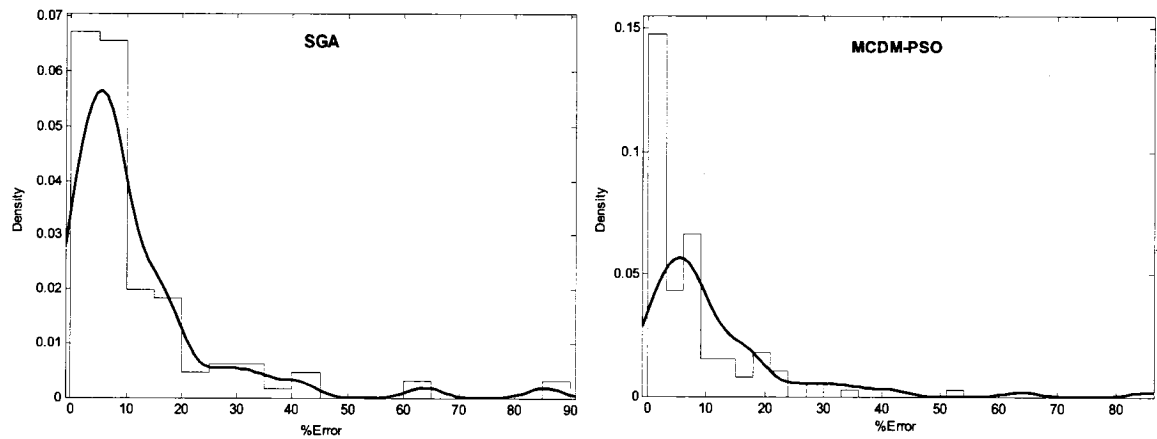


Figure 5.12: Nonparametric distribution fitting (average case).

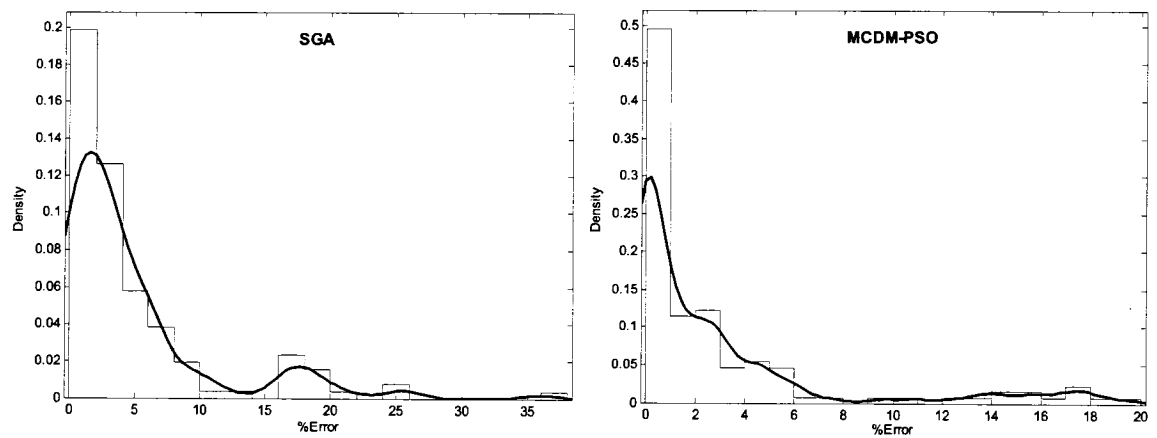


Figure 5.13: Nonparametric distribution fitting (best case)

The test is carried out using SPSS 14.0 and the results are given in the Tables (5.19-5.22). SPSS software provides two tables in the case of the nonparametric test for two independent samples: rank table, which contains mean and sum of ranks for each sample; and the other table contains statistics about the type of tests performed (in the cases of:  $U$  for Mann-Whitney,  $W$  for Wilcoxon and  $Z$  for normal approx. of  $U$ ). The value of  $Z$  gives a two-sided P-value less than 0.0001. This means that the difference between the two samples is highly significant and one can reject the null hypothesis that the two samples are drawn from the same population. Also, the test concludes that the apparent difference between the error gaps is really genuine and is not due to chance.

**Table 5.19: Ranks for average case samples**

	VAR00001	N	Mean Rank	Sum of Ranks
VAR00003	SGA	131	152.23	19942.00
	MCDM-PSO	131	110.77	14511.00
	Total	262		

**Table 5.20: Test Statistics for average case samples**

	VAR00003
Mann-Whitney U	5865.000
Wilcoxon W	14511.000
Z	-4.430
Asymp. Sig. (2-tailed)	.000

**Table 5.21 Ranks for best case samples**

	VAR00001	N	Mean Rank	Sum of Ranks
VAR00003	SGA	131	150.96	19775.50
	MCDM-PSO	131	112.04	14677.50
	Total	262		

**Table 5.22: Test Statistics for best case samples**

	VAR00003
Mann-Whitney U	6031.500
Wilcoxon W	14677.500
Z	-4.174
Asymp. Sig. (2-tailed)	.000

### 5.5 MCDM-PSO versus Traditional PSO

In this section, the technique is evaluated further by comparing its performance with that of the traditional PSO technique reported in reference [134]. Although the authors test only a few instances of QAP with relatively small size; their work is considered one of the rarest literatures that use PSO to solve the QAP. In their work, the main philosophy of the traditional PSO remains untouched. However, the authors extended the perception of both velocity and position from real values to fuzzy matrices. The authors also evaluated the performance of their technique against the ant colony optimization (ACO) for the studied QAP problems.

In this section, the technique is compared with the performance of the earlier mentioned PSO and also with the ant colony system reported in the same reference. To facilitate fair comparison, agents are set to be equal to the swarm size in the PSO and number of ants in the ACO. Table 5.23 shows the values of parameters used in comparison for both ACO and PSO.

**Table 5.23: Values of parameters used in case of PSO and ACO [134]**

Algorithm	Parameter name	Parameter value
ACO	Number of ants	5
	Weight of pheromone trail $\alpha$	1
	Weight of heuristic information $\beta$	5
	Pheromone evaporation parameter $\rho$	0.8
	Constant for pheromone updating Q	10
PSO	Swarm size	5
	Self-recognition coefficient c1	1.49
	Social coefficient c2	1.49
	Inertia weight w	0.9 : 0.1

For MCDM-PSO, 5 agents and 10 replications are used for each problem with a maximum of 1000 iterations. Results are averaged over ten runs and the mean value (AVG) and standard deviation (STD) in every case are reported in Table 5.24.

The MCDM-PSO reaches a better average than those of ACO and the traditional PSO over all the studied cases. However, the MCDM-PSO has a higher standard deviation than ACO and PSO in few cases. Higher standard deviations reflect lower stability that could be due to employing a small number of agents (5 agents). In general the performance of MCDM-PSO is superior to both ACO and PSO in most cases due to the diversity measure that has been considered in the technique.

**Table 5.24: MCDM-PSO versus ant colony (ACO) and traditional PSO**

Problem	ACO		PSO		MCDM-PSO	
	AVG	STD	AVG	STD	AVG	STD
Nug5	50.0	0	50.2	0.6325	50	0
Nug8a	222.8	3.9101	218.8	3.2931	214.4	1.26
Tai8a	85934	800.4784	83294	2698.1	78711	1251.6
Chr12a	16557	1661.6	13715	2098.0	10983	632.47
Tai12a	256180	3066.5	254230	5809.9	233030.4	8208.8
Chr20a	5438.8	261.3909	4456.0	389.8974	3173.8	325.2
Dre30	1849.6	82.1998	1592.0	118.4736	1087	72.9
Tho40	302840	3603.3	286670	5318.3	263312.6	4617.2
Tai50a	5626356	15225	5587622	52893	5342643.4	26634.9

To show the effect of the diversity criterion on the position update, the probability distribution of the Levenshtien edit distance for MCDM-PSO versus that of the traditional PSO is plotted after 100 iterations (Figures 5.14 and 5.15). The figures show that agents in the proposed technique are more diverse than particles of the traditional PSO even after the 100 iterations (both techniques are tested using a population of 100). This diversity helps the population to “live” longer by avoiding the state of stagnation and prevents premature convergence, the feature that gives the technique its uniqueness and conquers one of PSO drawbacks.

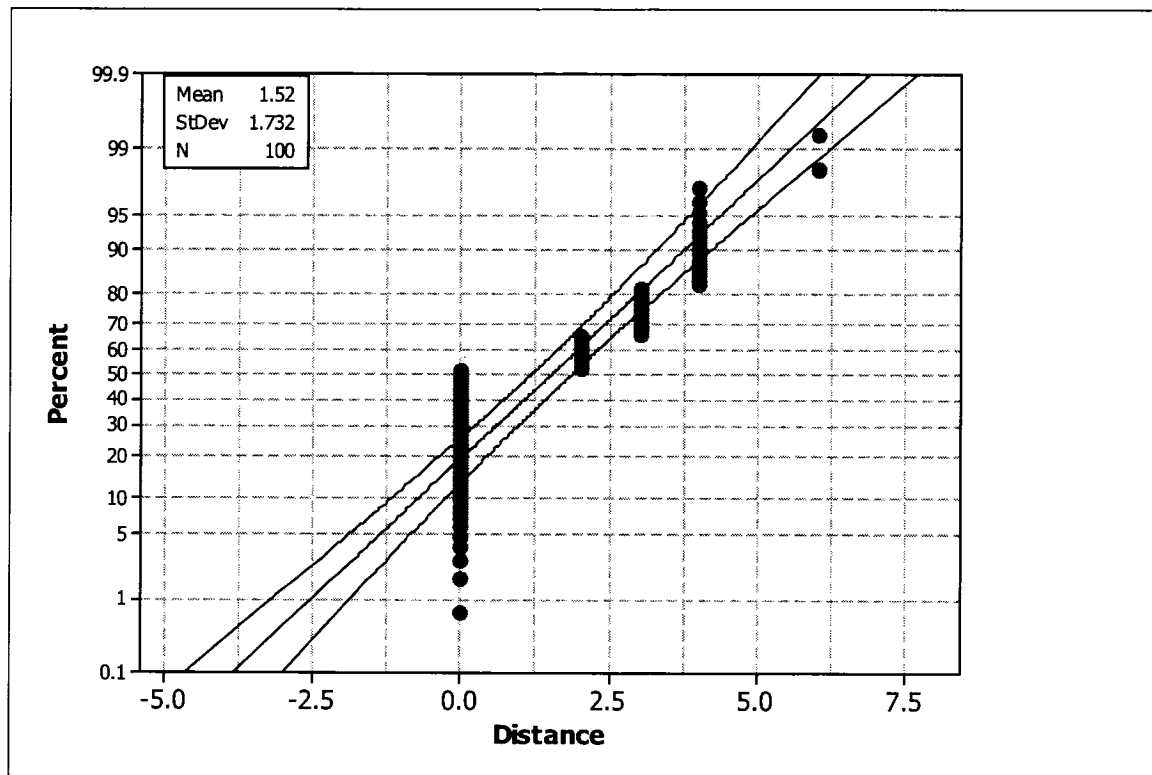


Figure 5.14: Probability Plot of Levenshtien distance matrix for traditional PSO after 100 iterations.



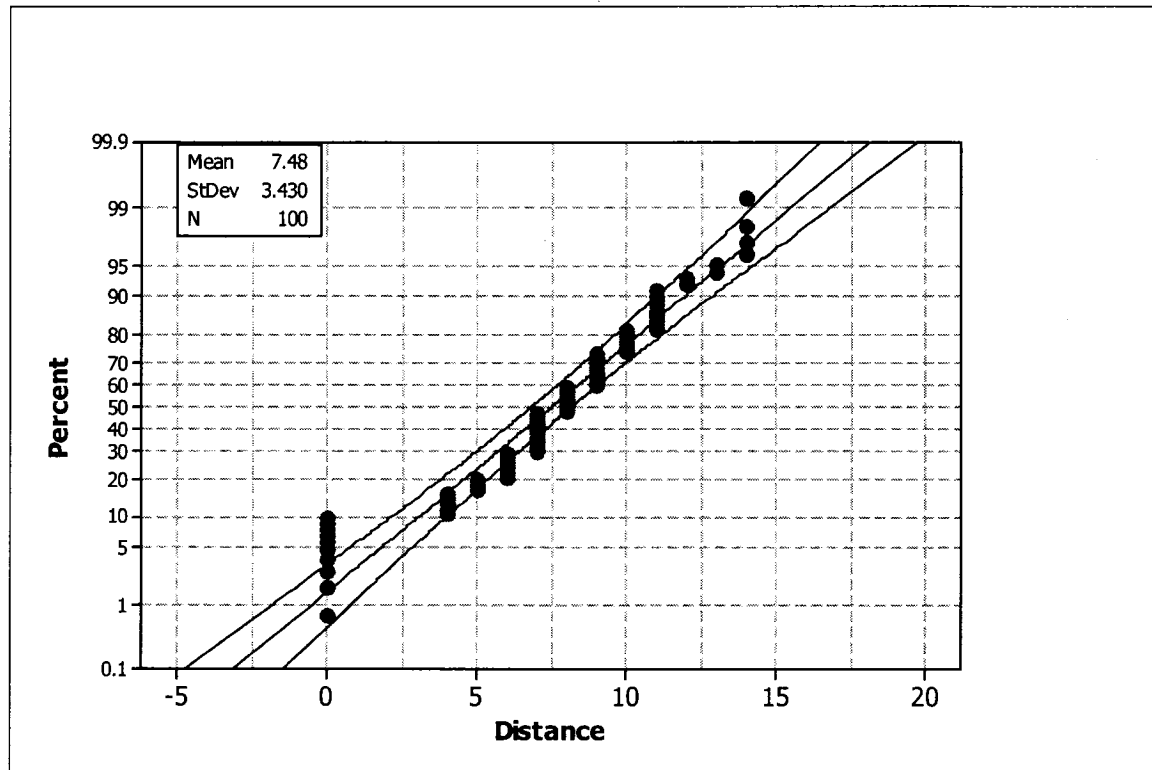


Figure 5.15: Probability Plot of Levenshtien distance matrix for MCDM-PSO after 100 iterations

## 5.6 Concluding remarks

For illustration, a step-by-step computation of the proposed technique was described with a moderate size example of TSP. The technique was also tested with a larger scale problem and was found to be effective in reaching a near optimal solution with both types of STP (symmetrical and Asymmetrical). Compared to traditional PSO, the proposed technique showed significant improvement in performance.

The quadratic assignment problem has a great deal of importance in the field of combinatorial optimization. New heuristics are usually evaluated according to its performance when used to solve QAP problems. The problem of QAP was extensively tested by selecting a wide range of problems to cover various applications. The results of simulation demonstrate the ability of MCDM-PSO to solve the problem with reasonable accuracy. When compared to the standard genetic algorithm, the technique solved the addressed problems with higher efficiency in both average and best cases. Moreover the technique also shows superiority over the traditional particle swarm due to the employed

diversity factor. To prove the significance of the error difference; the nonparametric Mann-Whitney test was used. The test is known to be powerful and effective when assumptions about normal distribution are violated, as in this case. The test results suggest higher significance of error gaps and rule out the possibility that the difference might be due to chance.

The availability of several alternatives for population agents facilitates higher discrepancies in genotypic positions and accordingly reduces the risk of premature convergence. The process of composing permutations using four alternatives is highly effective compared to the method of crossover used in genetic algorithms. Unlike crossover, the composition procedure relies on the objective mixing of four permutations while in crossover; permutations are generated using random sharing between two parents.

# Chapter 6: Extension to Continuous Variables Optimization

## 6.1 Introduction

Although the proposed technique is intended for combinatorial optimization, in this chapter the application to optimization problems with continuous variables is explored. Unlike the applications presented earlier, the variables in this case are not coded by a permutation of integers but rather by real values  $x \in R^n$  ( $R^n$  is the set of real numbers). Therefore the first significant change in the proposed MCDM-PSO would be the elimination of the Levenshtein edit distance from the algorithm. The swarm diversity is now expressed by the distance measured as the differences between the real values of the variables.

## 6.2 The MCDM-PSO model

As explained earlier in chapters 3 and 4, in the proposed technique, the position of an agent within the swarm is updated in accordance to its response to changes around it. Attitude (position) change is acquired by four main basic behaviors namely; imitation, memory retrieval, momentum, and play. So, the position update equation can be expressed as:

*New position = current position + momentum influence + memory retrieval influence + imitation influence + play influence.*

### 6.2.1 Momentum influence

The momentum is the tendency to continue the search in the current direction. In the experiment, several schemes for momentum are tried. However, random portions of the difference between the lower and upper limits of the variable are found to give the best results. To gain more control over the momentum influence, a momentum rate  $\eta$  is introduced to the position update equation so that the momentum influence is expressed as:

$$rand^{\eta}(X_{max} - X_{min}) \quad (6-1)$$

Where,

- $X_{min}$  and  $X_{max}$  are the minimum and maximum values allowed to the variable  $X$  respectively.
- $rand$  : is a uniformly distributed random number between 0 and 1.
- $\eta$  : is the momentum rate.

More details about tuning the parameter  $\eta$  are given in the section specifying MCDM-PSO setting.

### 6.2.2 Memory retrieval influence

The memory retrieval influence is a result of the relation between the agent's current state and its best experience. The magnitude of this influence is determined by the levels of fitness and diversity of the agent's best experience, which can be indicated by the memory retrieval ranking index  $RI_r$ . Both genotypic and phenotypic sides of that relation are given in Table 6.1.

**Table 6.1: key players in memory retrieval relation.**

	Genotypic side	Phenotypic side
Agent's current state	$X$	$X_{ft}$
Agent's best experience	$pbestx$	$pbest$

According to the proposed technique, the influence of memory retrieval should be weighted by the memory retrieval ranking index  $RI_r$ . This ranking index is a result of the multi-criterion decision making process explained previously in chapters 4 and 5. Therefore, the memory retrieval term of the position update equation can be expressed as:

$$RI_m(pbestx - X) \quad (6-2)$$

Where:

$$RI_m = \frac{RI_r}{RI_r + RI_i + RI_p} : \text{is the normalized memory retrieval ranking index.}$$

$RI_i$ : is the imitation ranking index.

$RI_p$ : is the play ranking index.

The second significant difference between the algorithm setting in the case of integer permutations and real valued variables is that the momentum influence is no longer included in the set of decision's alternatives as shown in the normalized value of the ranking index  $RI_r$ .

### 6.2.3 Imitation influence

Imitation is a one-sided interaction between an agent and the best agent in its neighborhood. Table 6.2 shows the key players engaged in imitation behavior.

**Table 6.2: key players for imitation influence.**

	Genotypic side	Phenotypic side
Agent's current state	$X$	$Xft$
State of the best agent in the neighborhood	$Nbestx$	$Nbest$

The imitation influence is also weighted by the imitation ranking index  $RI_i$  so that the imitation influence term in the position update equation can be expressed as:

$$RI_{in}(Nbestx - X) \quad (6-3)$$

Where:

$RI_{in} = \frac{RI_i}{RI_r + RI_i + RI_p}$  : is the normalized imitation ranking index.

#### 6.2.4 Play influence

As explained earlier in chapters 4 and 5, play behavior can be expressed by random moves in the domain space. However, total random moves could eventually have a negative impact on the convergence process and the unity of the swarm as a whole due to agents potentially wandering away from the swarm core. Therefore, a smart play action would be in a neighborhood surrounded by  $pbestx$ ,  $Nbestx$ , and  $gbestx$ . In the experiment this random position is created using a random normal distribution formula with mean equal to the average of those three points and standard deviation  $\mu$  so that:

$$Randx = normrnd\left(\frac{gbestx + Nbestx + pbestx}{3}, \mu\right) \quad (6-4)$$

Where;

$Randx$  : is the random position created to represent play behavior.

$normrnd$  : is a Matlab function returns random numbers chosen from a normal distribution.

$\left(\frac{gbestx + Nbestx + pbestx}{3}\right)$  : is the mean of the random numbers generator.

$\mu$  : is the standard deviation of the random numbers generator.

Details about tuning the parameter  $\mu$  are given in the section specifying MCDM-PSO settings.

Play influence on the position update equation is determined by the relation between the agent's current state and its random state (created by play behavior). This relation has two sides as shown in Table 6.3.

Table 6.3: Both sides of the play influence

	Genotypic side	Phenotypic side
Agent's current state	$X$	$X_{ft}$
Agent's random state	$Randx$	$Rndft$

Play influence is also weighted by the play ranking index  $RI_p$  so that the play influence term in the position update equation can be expressed as:

$$RI_{pn}(Randx - X) \quad (6-5)$$

Where:

$$RI_{pn} = \frac{RI_p}{RI_r + RI_i + RI_p} : \text{is the normalized play ranking index.}$$

### 6.2.5 Position update equation

Now the final form of the position update equation for a single agent in one dimension can be expressed as:

$$X^{k+1} = X^k + rand^\eta (Xmax - Xmin) + RI_m^k (pbestx^k - X^k) + RI_{in}^k (Nbestx^k - X^k) + RI_{pn}^k (Randx^k - X^k)$$

Where:

$k$ : is the iterations index.

The general form for the position update equation of an agent  $i$  in the dimension  $d$  would be:

$$X^{k+1}(i, d) = X^k(i, d) + rand^\eta * \{Xmax(d) - Xmin(d)\} + RI_m^k(i, d) * \{pbestx^k(i, d) - X^k(i, d)\} + RI_{in}^k(i, d) * \{Nbestx^k(i, d) - X^k(i, d)\} + RI_{pn}^k(i, d) * \{Randx^k(i, d) - X^k(i, d)\} \quad (6-6)$$

## 6.3 Computational experiments

To evaluate the performance of the proposed algorithm on optimizing real-valued problems, a large diverse set of benchmark functions is chosen from the literature [135-155]. The set of 32 benchmark problems contains well-known functions on the field of

unconstrained optimization that are widely used to test the functionality of heuristic techniques. The dimension of the test problems ranges from 2 to 100. Also the set contains several classes of problems including both unimodal and multimodal types with regular and irregular distribution of valleys and hills.

### 6.3.1 Benchmark problems

The proposed MCDM-PSO is tested using 22 functions ( $f_1 - f_{22}$ ). Functions  $f_1 - f_{10}$  are repeated in the test with 30 and 100 dimensions yielding a total of 32 test problems. Details of test problems including definitions, variable ranges, dimensionality, and optimal solutions are given below. Descriptions and visualizations of the problems are credited to references [135], [136] and [137].

#### 1. Sphere model

This is a unimodal function also known as De Jong's function 1. The definition of the function is:

$$f_1(x) = \sum_{d=1}^n x_d^2 \quad -5.12 \leq x_d \leq 5.12$$

The global minimum is  $f_1(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of the function in two different ranges is shown in Figure 6.1.

#### 2. Rotated hyper-ellipsoids function

The function is also unimodal and it produces a hyper-ellipsoid surface.

The definition of the function is

$$f_2(x) = \sum_{d=1}^n \left( \sum_{e=1}^d x_e \right)^2 \quad -65.536 \leq x_d \leq 65.536$$

The global minimum is  $f_2(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of the function in the range -50 to 50 is shown in Figure 6.2.

#### 3. Axis parallel hyper-ellipsoids

This function is a weighted sphere model, meaning it is also a unimodal function.

The definition of the function is:



$$f_3(x) = \sum_{d=1}^n d \cdot x_d^2 \quad -5.12 \leq x_d \leq 5.12$$

The global minimum is  $f_3(x) = 0$ ;  $x_d = 0$ ,  $d = 1:n$

Visualization of the function in the range -5 to 5 is shown in Figure 6.3.

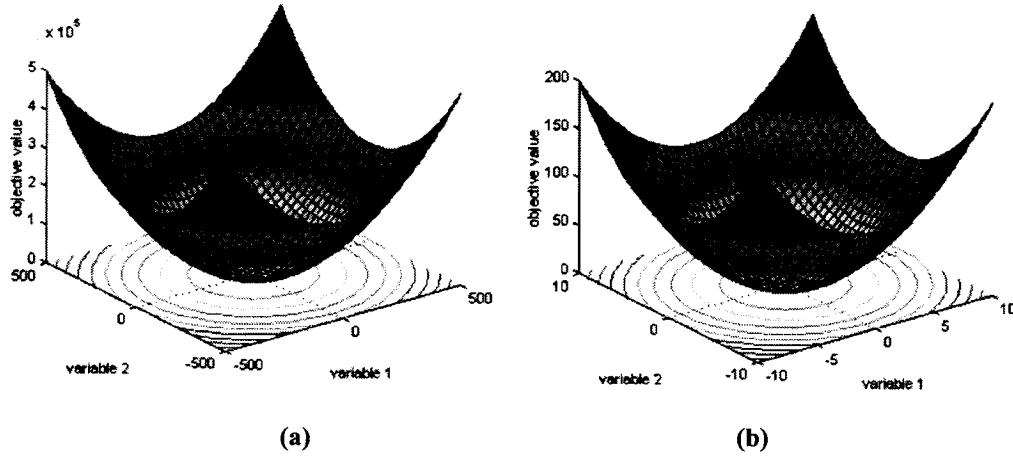


Figure 6.1: Visualization of the Sphere model;

(a) In the range -500 to 500 and (b) A closer look in the range -10 to 10

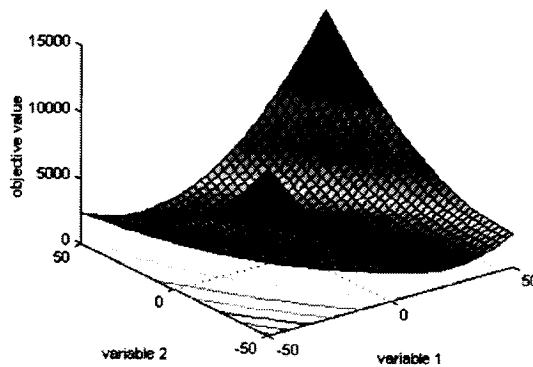


Figure 6.2: Visualization of the Rotated hyper-ellipsoids function

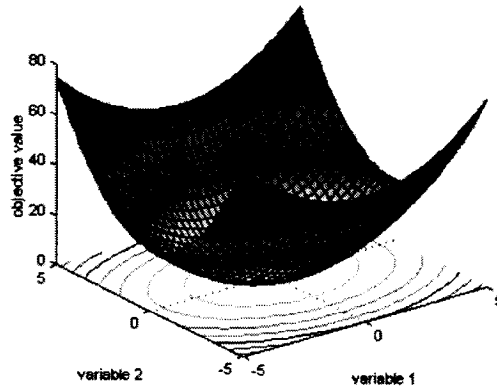


Figure 6.3: Axis parallel hyper-ellipsoids

#### 4. Rastrigin function

This function is based on the sphere model except for the cosine term that produces many local minima. The function is highly multimodal with regularly distributed local minima. The definition of the function is:

$$f_4(x) = \sum_{d=1}^n (x_d^2 - 10\cos(2\pi x_d) + 10) \quad -5.12 \leq x_d \leq 5.12$$

The global minimum is  $f_4(x) = 0$ ;  $x_d = 0$ ,  $d = 1:n$

Visualization of the function in two different ranges is shown in Figure 6.4.

#### 5. Schwefel function

This is a multimodal function with many local optima. The complexity of this function is due to the fact that the global optimal is far from the local optima, which makes it hard to optimize if many agents fall into one of the deep local optima [138]. The definition of this function is:

$$f_5(x) = \sum_{d=1}^n -x_d \cdot \sin(\sqrt{|x_d|}) \quad -500 \leq x_d \leq 500$$

The global minima is  $f_5(x) = -n.418.9829$ ;  $x_d = 420.9687$ ,  $d = 1:n$

Visualization of this function in the full range is shown in Figure 6.5.

### 6. Rosenbrock valley function

This function is also known as the banana function. The global minimum of this function is located inside a long, narrow, parabolic shaped flat valley with many deceptive local minima. It is easy to find the valley but very hard to converge to the global optimum. The definition of this function is:

$$f_6(x) = \sum_{d=1}^{n-1} [100(x_{d+1} - x_d^2)^2 + (1 - x_d)^2] \quad -2.048 \leq x_d \leq 2.048$$

The global minima is  $f_6(x) = 0$ ;  $x_d = 1$ ,  $d = 1:n$

Visualization of this function in two different ranges is shown in Figure 6.6.

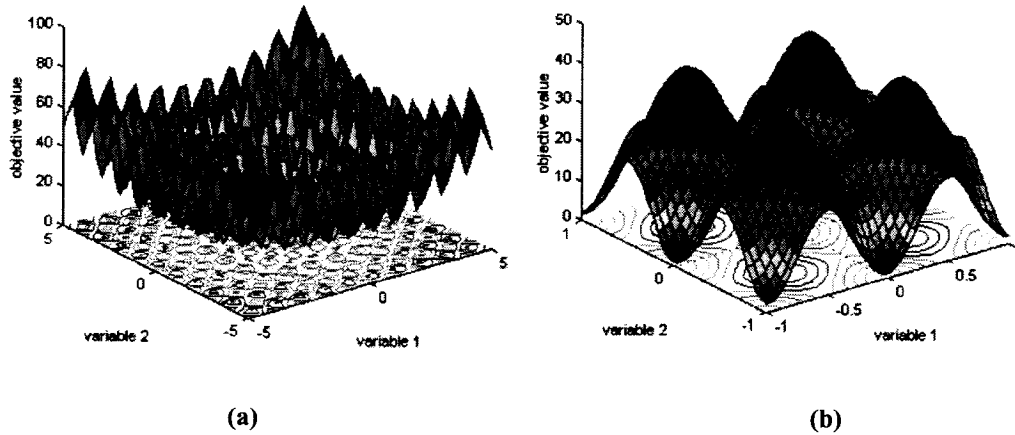


Figure 6.4: Visualization of the Rastrigin function; (a) between -5 and 5 and (b) between -1 and 1.

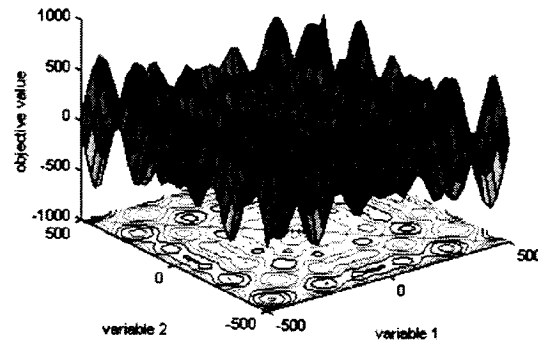


Figure 6.5: Visualization of the Schwefel function.

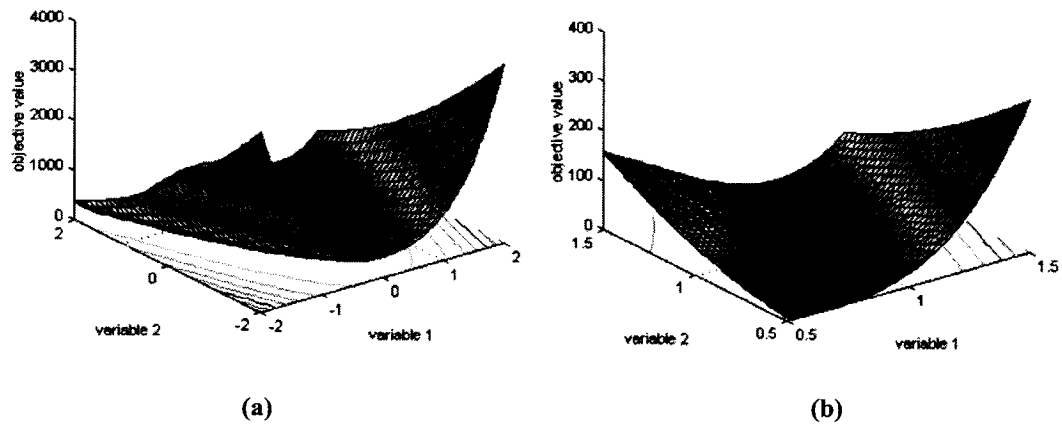


Figure 6.6: Visualization of the Rosenbrock function; (a) between -2 and 2 and (b) around the local minimum (between -0.5 and 1.5).

### 7. Griewangk function

This is a multimodal function with regularly distributed local minima. The definition of this function is:

$$f_7(x) = \sum_{d=1}^n \frac{x_d^2}{4000} - \prod_{d=1}^n \cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \quad -600 \leq x_d \leq 600$$

The global optimum is  $f_7(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of this function in three different ranges is shown in Figure 6.7.

### 8. Ackley path function

This function is multimodal, with a global optimum located in a narrow basin and surrounded by several shallow local minima. The definition of the function is:

$$f_8(x) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{d=1}^n x_d^2}} - e^{-\frac{1}{n}\sum_{d=1}^n \cos(2\pi x_d)} \quad -32.768 \leq x_d \leq 32.768$$

The global optimum is  $f_8(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of this function in two different ranges is shown in Figure 6.8.

### 9. The sum of different powers function

This is a unimodal function. The definition of the function is:

$$f_9(x) = \sum_{d=1}^n |x_d|^{(d+1)} \quad -1 \leq x_d \leq 1$$

The global minima is  $f_9(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of the function is shown in Figure 6.9.

### 10. Zakharov function

This is also a unimodal function. The definition of the function is:

$$f_{10}(x) = \sum_{d=1}^n x_d^2 + \left( \sum_{d=1}^n 0.5 \cdot d \cdot x_d \right)^2 + \left( \sum_{d=1}^n 0.5 \cdot d \cdot x_d \right)^4 \quad -5 \leq x_d \leq 10$$

The global solution is  $f_{10}(x) = 0$ ;  $x_d = 0$ ,  $d = 1 : n$

Visualization of the function is shown in Figure 6.10.

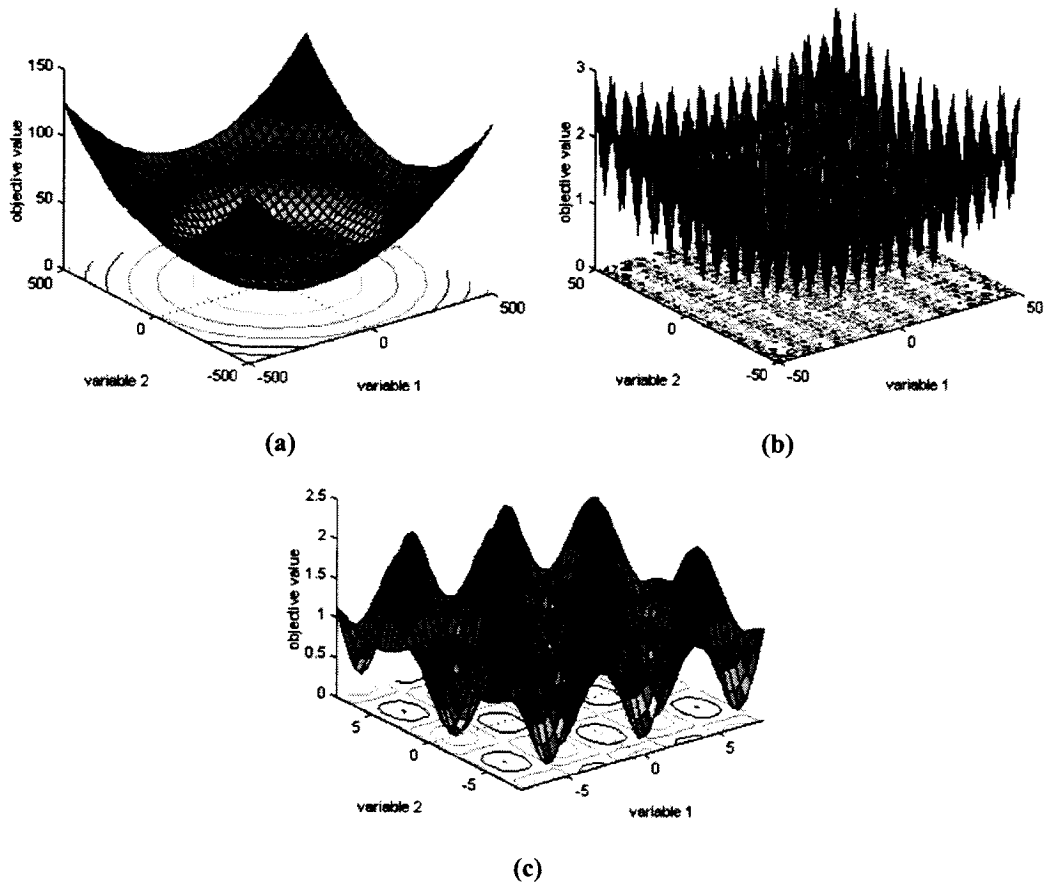


Figure 6.7: Visualization of the Griewangk function; (a) in the full range, (b) between -50 and 50 and (c) between -8 and 8.

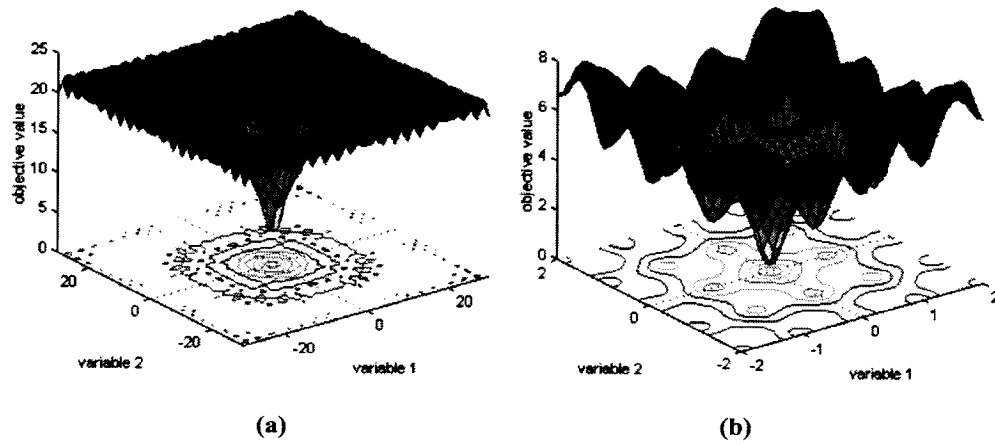


Figure 6.8: Visualization of Ackley path function; (a) between -30 and 30 and (b) between -2 to 2.

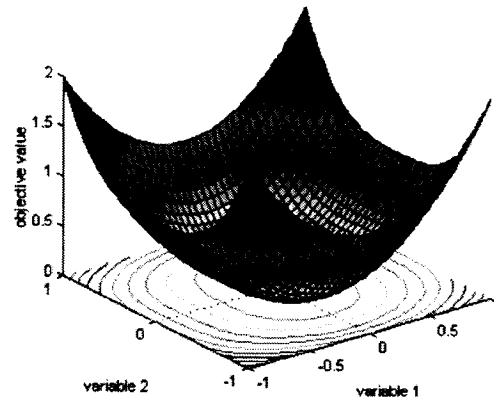


Figure 6.9: Visualization of the sum of different powers function.

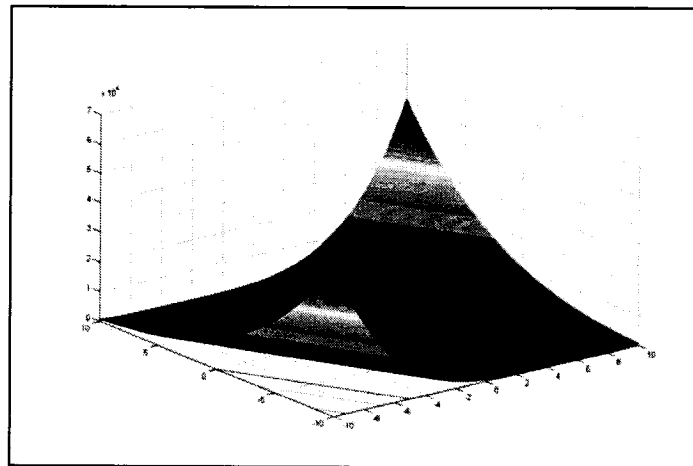


Figure 6.10: Visualization of the Zakharov function.

### 11. Langerman function

This is a multimodal function. The difficulty of this function is due to the roughness of its landscape where its global minimum is located within several unevenly distributed local minima. The definition of this function is:

$$f_{11}(x) = -\sum_{d=1}^5 c_d \left( e^{-\frac{1}{\pi} \|X - A(1:5, d)\|^2} \cdot \cos(\pi \|X - A(1:5, d)\|^2) \right)$$

$$X = [x_{1:d}]^T \quad 0 \leq x_d \leq 10; \quad d = 1:5$$

$$A = \begin{bmatrix} 9.6810 & 9.4000 & 8.0250 & 2.1960 & 8.0740 \\ 0.6670 & 2.0410 & 9.1520 & 0.4150 & 8.7770 \\ 4.7830 & 3.7880 & 5.1140 & 5.6490 & 3.4670 \\ 9.0950 & 7.9310 & 7.6210 & 6.9790 & 1.8630 \\ 3.5170 & 2.8820 & 4.5640 & 9.5100 & 6.7080 \end{bmatrix}; \quad C = \begin{bmatrix} 0.8060 \\ 0.5170 \\ 1.5000 \\ 0.9080 \\ 0.9650 \end{bmatrix}$$

The best value to reach for this function is defined by the International Contest on Evolutionary Optimization ICEO [139] to be  $f_{11}(x) = 1.4$ .

Visualization of this function with two different sets of variables for the same range is shown in Figure 6.11.

### 12. Michalewicz function

This is a multimodal function with several local minima. The definition of the function is:

$$f_{12}(x) = -\sum_{d=1}^5 \sin(x_d) \cdot \sin^{2m} \left( \frac{d \cdot x_d^2}{\pi} \right); \quad 0 \leq x_d \leq \pi; \quad d = 1:5, \quad m = 10$$

The best value to reach for this function is defined by the International Contest on Evolutionary Optimization ICEO [139] to be  $f_{12}(x) = -4.6876$ .

Visualization of this function with two different sets of variables in different ranges is shown in Figure 6.12.



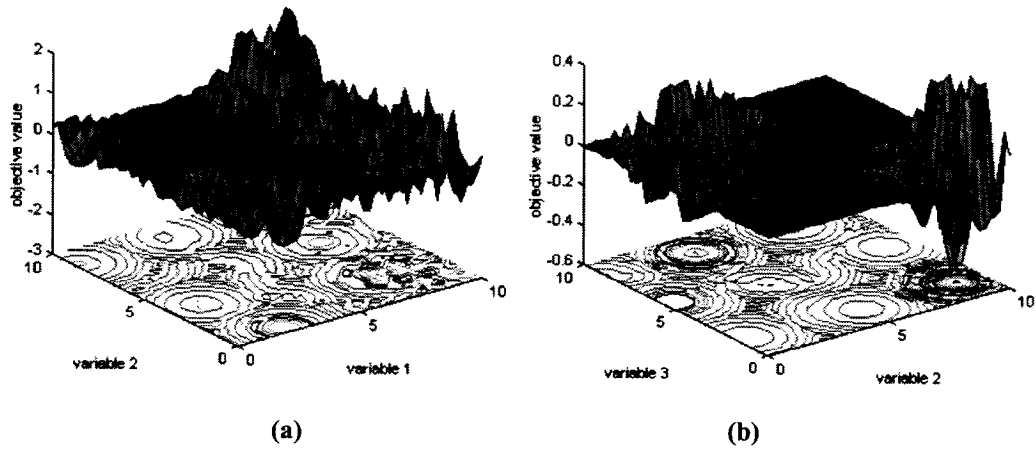


Figure 6.11: Visualization of the Langerman function; (a) for the first and second variables (b) for the second and third variables with first variable set to zero

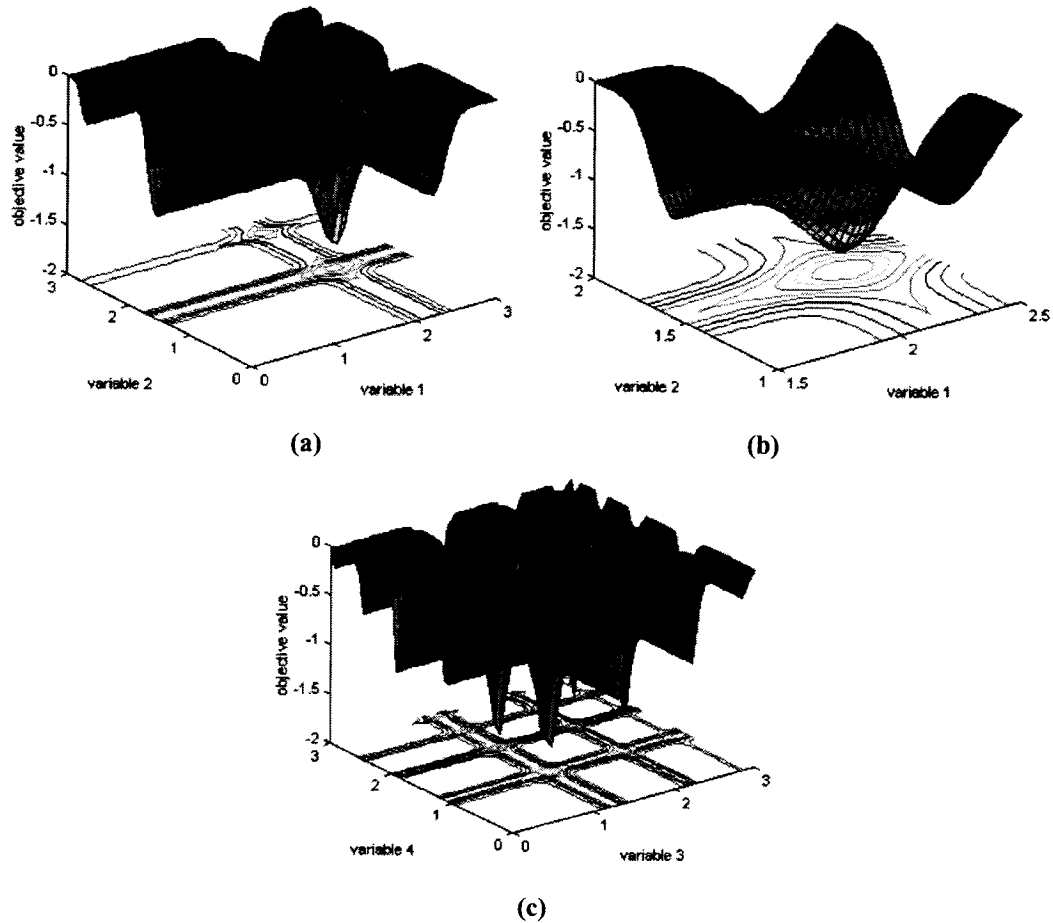


Figure 6.12: Visualization of the Michalewicz function; (a) with variable 1 and variable 2 in the range 0 to 3, (b) in the range 1.5 to 2.5 and (c) with variables 3 and 4 while variables 1 and 2 are set to zero in the rang 0 to 3.

**13. Easom function**

This is a unimodal function. The definition of the function is:

$$f_{13}(x) = -\cos(x_1) \cdot \cos(x_2) \cdot e^{-((x_1 - \pi)^2 + (x_2 - \pi)^2)}; \quad -100 \leq x_d \leq 100; \quad d = 1 : 2$$

The global solution is  $f_{13}(x) = -1$ ;  $x_d = \pi$ ,  $d = 1 : 2$

Visualization of this function in two different ranges is shown in Figure 6.13.

**14. Six-hump camel back function**

This is a multimodal test function with two global minima located among the total of six local minima. The definition of the function is:

$$f_{14}(x) = (4 - 2.1x_1^2 + x_1^{4/3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2; \quad -3 \leq x_d \leq 3; \quad d = 1 : 2$$

The global minimum is  $f_{14}(x) = -1.0316$ ;  $(x_1, x_2) = (-0.0898, 0.7126), (0.0898, -0.7126)$

Visualization of this function in two different scales is shown in Figure 6.14.

**15. Branin function**

This function has no local minima and two global minima in the specified range. The definition of the function is:

$$f_{15}(x) = (x_2 - \frac{5.1}{4\pi}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10 \cdot (1 - \frac{1}{8\pi}) \cdot \cos(x_1) + 10 \quad 0 \leq x_d \leq 10; \quad d = 1 : 2$$

The global minimum is  $f_{15}(x) = 0.397887$ ;  $(x_1, x_2) = (\pi, 2.275), (9.42478, 2.475)$ .

Visualization of this function is shown in Figure 6.15.

**16. Hump function**

This is the same as function 14 except that the global solution is equal to zero. The definition of the function is:

$$f_{16}(x) = 1.0316285 + 4x_1^2 - 2.1x_1^4 + x_1^{6/3} + x_1x_2 - 4x_2^2 + 4x_2^4; \quad -5 \leq x_d \leq 5, \quad d = 1 : 2$$

The global solution is  $f_{16}(x) = 0$ ;  $(x_1, x_2) = (0.0898, -0.7126), (-0.0898, 0.7126)$

Visualization of this function is shown in Figure 6.16.

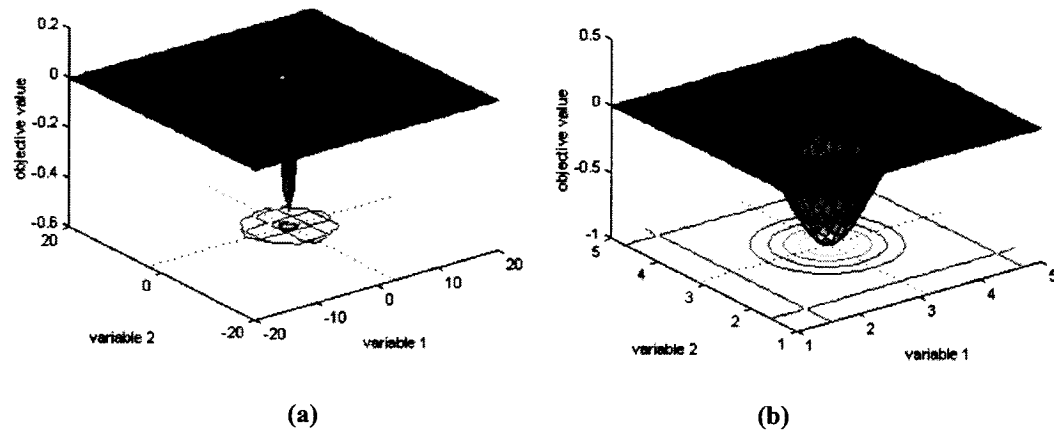


Figure 6.13: Visualization of the Easom function; (a) between -20 and 20 and (b) between 1 and 5.

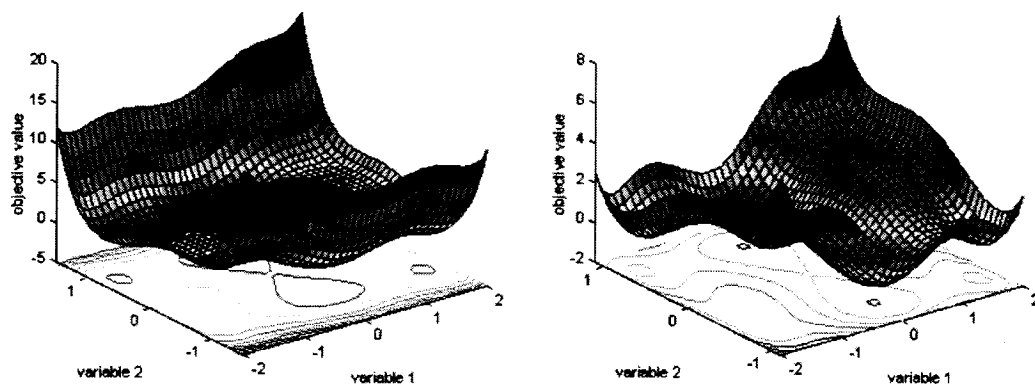


Figure 6.14: Visualization of the Six-hump camel back function; (a) wide overview (b) focused view around the minima

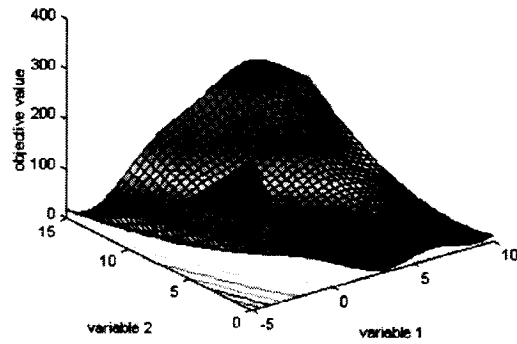


Figure 6.15: Visualization of the Branin function.

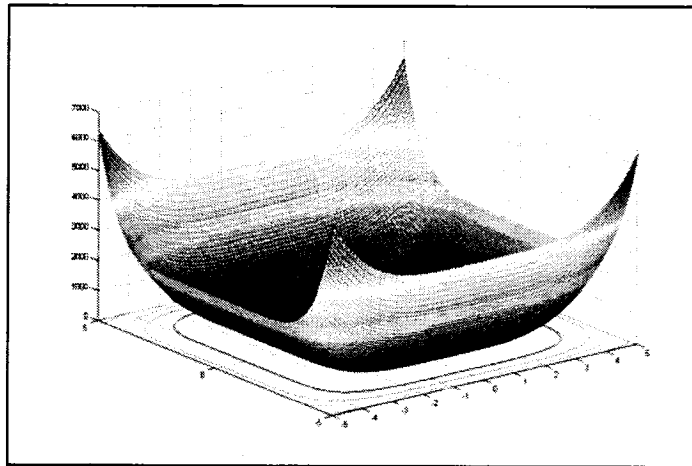


Figure 6.16: Visualization of the Hump function.

**17. Hartmann 3-dimentional function**

This is a multimodal function with four local minima; one of them is the global solution. The definition of the function is:

$$f_{17}(x) = -\sum_{i=1}^4 \alpha_i \exp\left[-\sum_{d=1}^3 A_{id}(x_d - P_{id})^2\right]; \quad 0 \leq x_d \leq 1, \quad d = 1:3$$

$$\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]^T; \quad A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}; \quad P = 10^{-4} \begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$

The global minimum is  $f_{17}(x^*) = -3.86278$ ,  $x^* = (0.114614, 0.555649, 0.852547)$

**18. Hartmann 6-dimentional function**

This is a multimodal function with six local minima; one of them is the global solution. The definition of the function is:

$$f_{18}(x) = -\sum_{i=1}^6 \alpha_i \exp\left[-\sum_{d=1}^6 B_{id}(x_d - Q_{id})^2\right]; \quad 0 \leq x_d \leq 1, \quad d = 1:6$$

$$\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]^T, \quad B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad Q = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

The global minimum is:

$$f_{18}(x^*) = -3.32237; \quad x^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573),$$

**19. Shekel function**

This is a multimodal function with  $m$  local minima; one of them is the global solution. The definition of the function is:

$$f_{19}(x) = -\sum_{i=1}^m \left[ \sum_{d=1}^4 (x_d - C_{di})^2 + \beta_i \right]^{-1}; \quad m = 10, \quad 0 \leq x_d \leq 10, \quad d = 1:4$$

$$\beta = \frac{1}{10} [1 \quad 2 \quad 2 \quad 4 \quad 4 \quad 6 \quad 3 \quad 7 \quad 5 \quad 5]^T, \quad C = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}$$

The global minimum is  $f_{19}(x^*) = -10.5364$ ,  $x^* = (4, 4, 4, 4)$

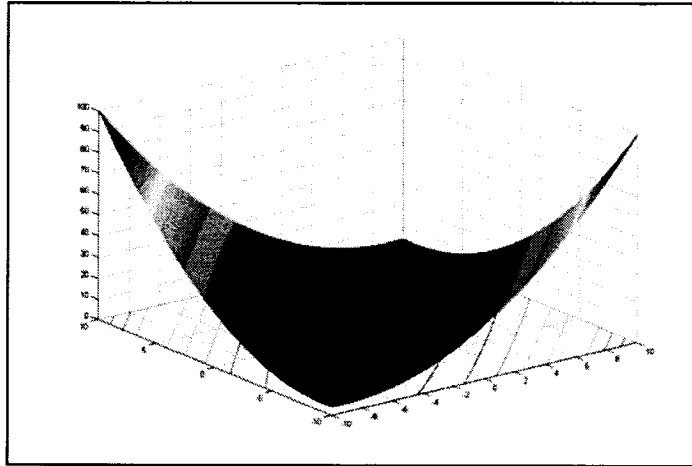


Figure 6.17: Visualization of the Matyas function.

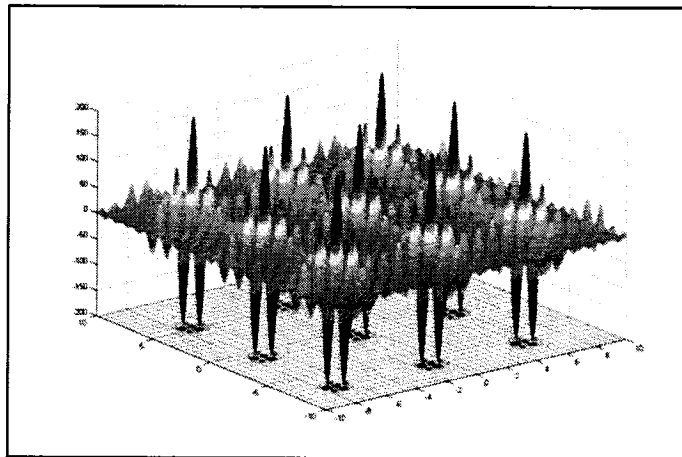
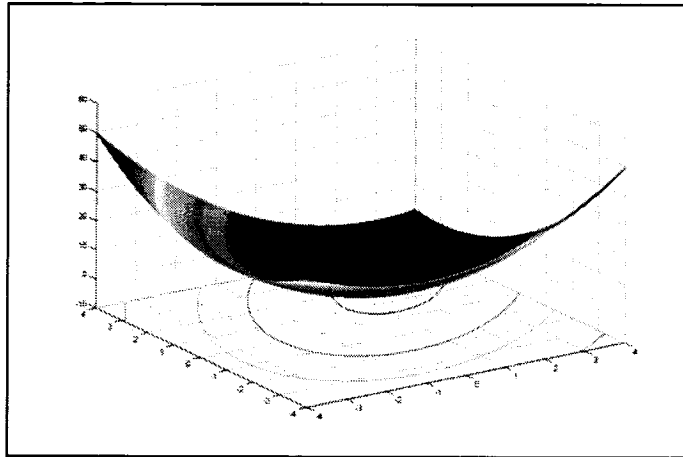


Figure 6.18: Visualization of the Shubert function.



**Figure 6.19: Visualization of the Trid function.**

### 20. Matyas function

This is a unimodal function. The definition of the function is:

$$f_{20}(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2; \quad -10 \leq x_d \leq 10, \quad d = 1:2$$

The global minimum is  $f_{20}(x^*) = 0$ ,  $x^* = (0, 0)$ ,

Visualization of this function is shown in Figure 6.17.

### 21. Shubert function

This is a highly multimodal function with 18 global minima. The definition of the function is:

$$f_{21}(x) = \left( \sum_{i=1}^5 i \cos((i+1)x_1 + i) \right) \left( \sum_{i=1}^5 i \cos((i+1)x_2 + i) \right); \quad -10 \leq x_d \leq 10, \quad d = 1:2$$

The function has 18 global minima with  $f_{21}(x^*) = -186.7309$

Visualization of this function is shown in Figure 6.18.

### 22. Trid function

This is a unimodal function. The definition of the function is:

$$f_{22}(x) = \sum_{d=1}^n (x_d - 1)^2 - \sum_{d=2}^n x_d x_{d-1}; \quad -n^2 \leq x_d \leq n^2, \quad d = 1:n$$

The global minimum  $n=6$  is  $f_{22}(x^*) = -50$  at  $x_d^* = d(7-d)$ ,  $d = 1:6$

The global minimum  $n=10$  is  $f_{22}(x^*) = -210$  at  $x_d^* = d(11-d)$ ,  $d = 1:10$

Visualization of this function is shown in Figure 6.19.

## 6.3.2 Test environment and MCDM-PSO setting

The MCDM-PSO for continuous optimization is coded on the Matlab 7 platform and is run on an Intel Pentium M7255 machine. Test results are averaged over 30 runs for problems  $f_1$  to  $f_{10}$  and over 20 runs for problems  $f_{11}$  to  $f_{22}$ . The mean and standard deviations of the solutions are recorded. Although a maximum of 2000 iterations is set for every run, in most cases the MCDM-PSO is able to find reasonable solutions before reaching the predefined maximum limit of iterations. In these cases, the number of



iterations other than the maximum limit is also recorded. A population size of 30 agents is used for all problems. A neighborhood size of 30 is found to be the best setting for problems with dimensions of 30 and 100, whereas a neighborhood size of 5 is suitable for problems with dimensions less than 30. Positions of agents are randomly initialized between the given ranges of variables for each problem. The value of the momentum rate parameter  $\eta$  is manually tuned for each problem individually. While a very high value for  $\eta$  could slow down the convergence, very low value causes big jumps in the search landscape. Taking big jumps in the search space could lead the agents to miss rich sub-areas or causes a cancellation of the position update step altogether because the position in the proposed algorithm is set to the agent's best position in the case of the boundary violation. Typical values for the parameter  $\eta$  are 1, 10, 100, 1000, etc. For the play behavior, the value of the parameter  $\mu$  is manually tuned for each problem individually according the roughness of the search landscape and the distribution of the valleys and hills on the surface. For a rough surface with irregular distribution of local optima, a small value of  $\mu$  is preferred. It also controls the level of the desired quality of search. For an intimate search around the best areas, a small value of  $\mu$  should be chosen. The value of  $\mu$  is also preferred to be equal or less than the upper limit of the variable ( $X_{max}$ ) to ensure that agents stay inside the domain. Generally, small values of  $\mu$  will not harm the quality of search but could significantly slow down the speed of convergence. From the experiments, typical values for the parameter  $\mu$  are found to be 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, etc.

### 6.3.3 Algorithms used for comparison

To gain more sense about the performance of the proposed technique, the results are compared with results obtained by the traditional PSO and six other well-established PSO variants that are previously proposed to overcome the problem of the premature convergence in the PSO especially with multimodal problems. Also, the results are compared with results obtained by an algorithm outside the PSO family which is simple evolutionary algorithm (SEA). Namely, algorithms involved in the comparison are:

1. Traditional particle swarm optimizer PSO.

2. Attractive-repulsive particle swarm optimizer ARPSO [140].
3. Unified particle swarm optimizer UPSO [141].
4. Fitness-distance ratio based particle swarm optimizer FDR-PSO [142].
5. Fully informed particle swarm optimizer FIPS [143].
6. Hybrid Cooperative particle swarm optimizer CPSO-H [144].
7. Comprehensive learning particle swarm optimizer CLPSO [138].
8. Simple evolutionary algorithm SEA [145].

## 6.4 Test results

### 6.4.1 MCDM-PSO vs. traditional PSO

For the comparison, the traditional PSO with decreasing inertia weight is used. The inertia weight function is decreased as the iteration index increases as:

$$W = rand^k (Xmax - Xmin)$$

For both MCDM-PSO and traditional PSO, positions are set to the agent/particle self best positions if the candidate move is found to be outside the search domain.

To have a fair comparison, the number of iterations in the case of traditional PSO is set to be equal to those needed to reach solutions recorded for MCDM-PSO and shown in Tables 6.4, 6.5 and 6.6. Also a population size of 30 is used for the traditional PSO.

First, both MCDM-PSO and traditional PSO are tested with problems  $f_1 - f_{10}$  on 30 and 100 dimensions. The mean and the standard deviations of the results in the case of 30-dimensional and 100-dimensional problems are presented in Tables 6.4 and 6.5 respectively. The best solutions are shown in boldface in the tables.

Results show that the proposed MCDM-PSO is able to quickly converge to very good solutions for the specified test problems (in most cases 100 iterations are recorded). Also, results show that MCDM-PSO significantly outperforms the traditional PSO in all test problems of 30 and 100 dimensions. Moreover, the consistency (the ability to find the same solution every run) of MCDM-PSO is very high due to the very small deviations recorded over the 30 runs as shown in the tables. Convergence characteristics of MCDM-PSO and traditional PSO with problems  $f_1 - f_{10}$  on 30 dimensions are shown in Figures 6.20 to 6.29.

**Table 6.4: Results for problems  $f_1$  to  $f_{10}$  on 30 dimensions.**

No.	Function Name	Iterations	MCDM-PSO		PSO	
			Mean	Std.	Mean	Std.
1	Sphere	100	<b>1.3946996e-52</b>	2.10e-53	1.65755	0.72711
2	Rotated hyper-ellipsoids	100	<b>2.9970622e-75</b>	8.16e-76	53.592529	12.90
3	Axis parallel hyper-ellipsoids	100	<b>1.5608442e-76</b>	5.02e-77	3.2740e+2	1.45e+2
4	Rastrigin	100	<b>0</b>	0	1.6106e+2	36.08
5	Schwefel	500	<b>-1.2569443e+4</b>	0.06757	-1.253e+4	1.45e+2
6	Rosenbrock	1000	<b>6.3956471e-5</b>	1.00e-4	51.93666	29.3065
7	Griewangk	500	<b>1.9613940e-16</b>	2.74e-16	0.750479	0.23447
8	Ackley	100	<b>8.8817841e-16</b>	0	8.912487	1.703
9	The sum of different powers	100	<b>8.0827494e-77</b>	6.94e-77	9.334e-6	1.36e-5
10	Zakharov	100	<b>5.2377566e-74</b>	2.11e-73	37.64920	37.6492

Table 6.5: Results for problems  $f_1$  to  $f_{10}$  on 100 dimensions

No.	Function Name	Iterations	MCDM-PSO		PSO	
			Mean	Std.	Mean	Std.
1	Sphere	100	<b>3.902412e-052</b>	3.63e-053	1.103111e+2	21.98
2	Rotated hyper-ellipsoids	100	<b>1.078668e-74</b>	1.82e-75	7.036120e+2	1.27e+2
3	Axis parallel hyper-ellipsoid	100	<b>1.610309e-75</b>	2.80e-76	2.206017e+5	4.89e+4
4	Rastrigin	100	<b>0</b>	0	9.005449e+2	97.38
5	Schwefel	500	<b>-4.1898185e+4</b>	0.2628846	-4.12981e+4	4.68e+2
6	Rosenbrock	1000	<b>1.6732261e-4</b>	2.15e-4	6.85716e+2	1.00e+2
7	Griewangk	500	<b>7.0314124e-17</b>	1.21e-16	53.41185543	16.676593
8	Ackley	100	<b>8.8817841e-16</b>	0	17.98391	0.45909
9	The sum of different powers	100	<b>6.8741940e-77</b>	4.85e-077	3.9400e-4	4.65e-4
10	Zakharov	100	<b>3.6758269e-74</b>	1.05e-74	1.25867e+3	1.69e+2

Second, the MCDM-PSO and traditional PSO are tested with problems less than 30 dimensions ( $f_{11}$  -  $f_{22}$ ) (Table 6.6). For this group of problems both MCDM-PSO and PSO are performing almost the same except for a few cases. For example, for the problem  $f_{19}$  the MCDM-PSO has a significantly better result than PSO. However, for the problem  $f_{20}$  the PSO is able to find better minimum than MCDM-PSO. Other than that, solutions obtained by both algorithms for that group are very slightly different. The convergence characteristics of both algorithms on the problem  $f_{20}$  are shown in Figure 6.30. The figure shows that MCDM-PSO and PSO have almost the same pattern of convergence with that particular problem.

**Table 6.6: Results for problems less than 30 dimensions.**

No.	Function Name	Iterations	MCDM-PSO		PSO	
			Mean	Std.	Mean	Std.
11	Langerman	1000	-1.367	0.2855	<b>-1.377</b>	0.2548
12	Michalewicz	2000	-4.622	0.0900	<b>-4.680</b>	0.0335
13	Easom	500	<b>-1</b>	0	-0.95	0.2236
14	Six-hump	1000	<b>-1.032</b>	3.94e-5	<b>-1.032</b>	0
15	Branin	1000	<b>0.397</b>	2.56e-5	<b>0.397</b>	0
16	Hump	1000	5.584e-5	5.11e-5	<b>4.651e-8</b>	0
17	Hartmann	1000	<b>-3.862</b>	5.87e-6	<b>-3.862</b>	2.2e-15
18	Hartmann	1000	<b>-3.302</b>	0.0448	-3.274	0.0599
19	Shekel	1000	<b>-10.536</b>	2.54e-5	-6.129	2.32
20	Matyas	1000	6.076e-13	4.9e-13	<b>3.748e-030</b>	1.1e-29
21	Shubert	1000	<b>-1.867e+2</b>	0.0120	<b>-1.867e+2</b>	3.8e-14
22	Trid	2000	<b>-2.099e+2</b>	0.0062	-2.098e+2	0.143

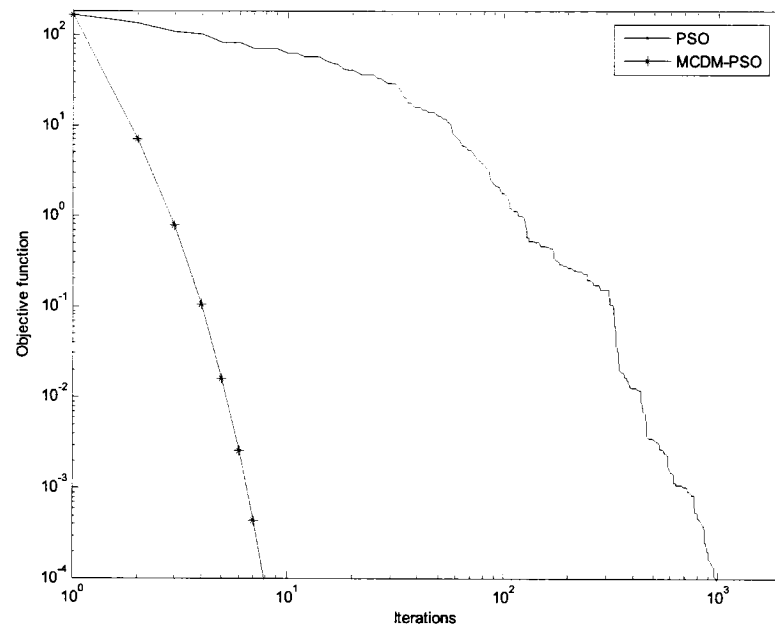


Figure 6.20: Convergence of the 30 dimensions Sphere function.

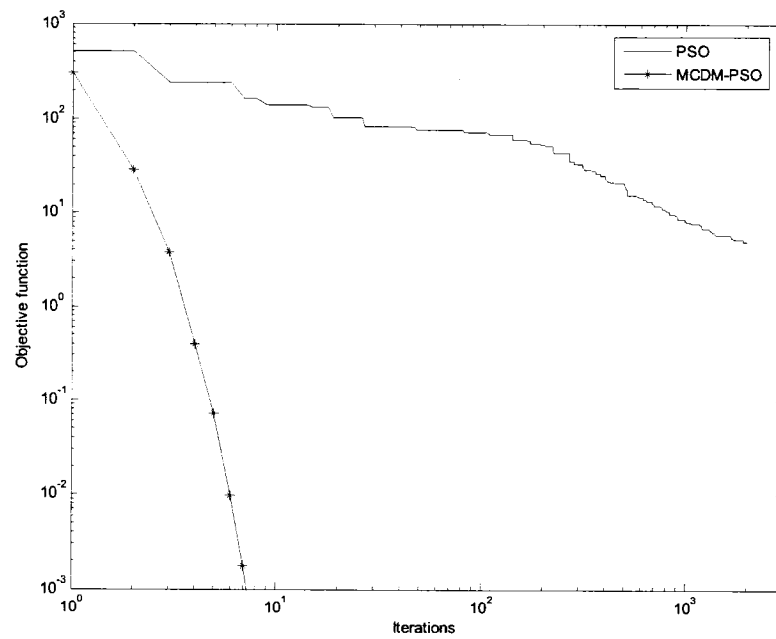


Figure 6.21: Convergence of the 30 dimensions Rotated hyper-ellipsoids function.

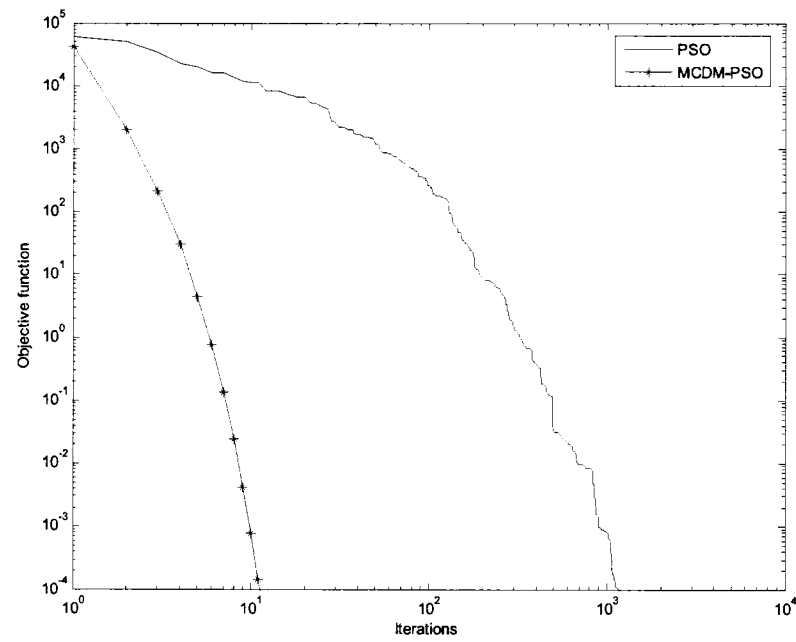


Figure 6.22: Convergence of the 30 dimensions Axis parallel hyper-ellipsoids function.

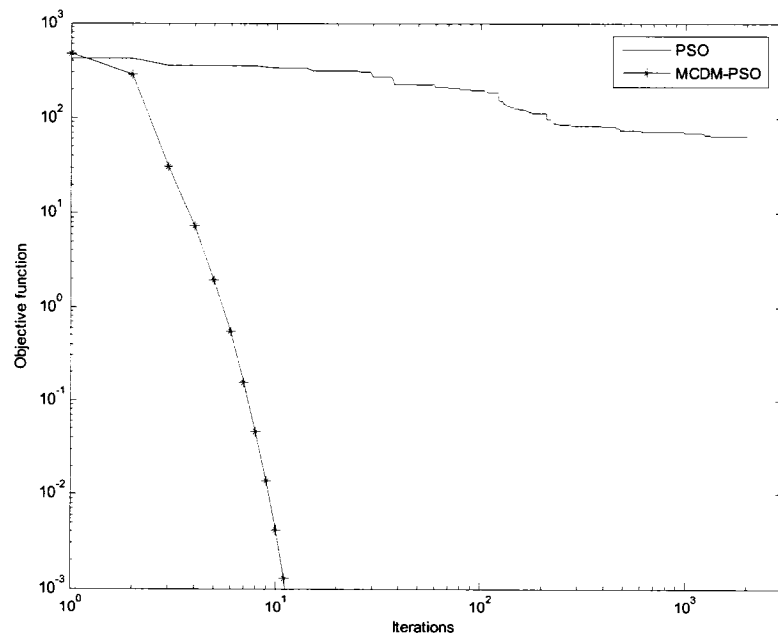


Figure 6.23: Convergence of the 30 dimensions Rastrigin function.

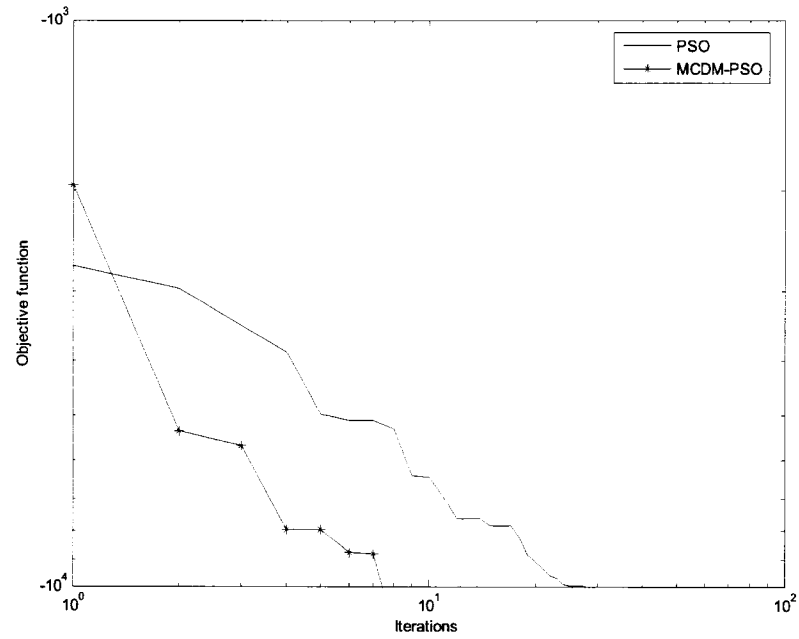


Figure 6.24: Convergence of the 30 dimensions Schwefel function.

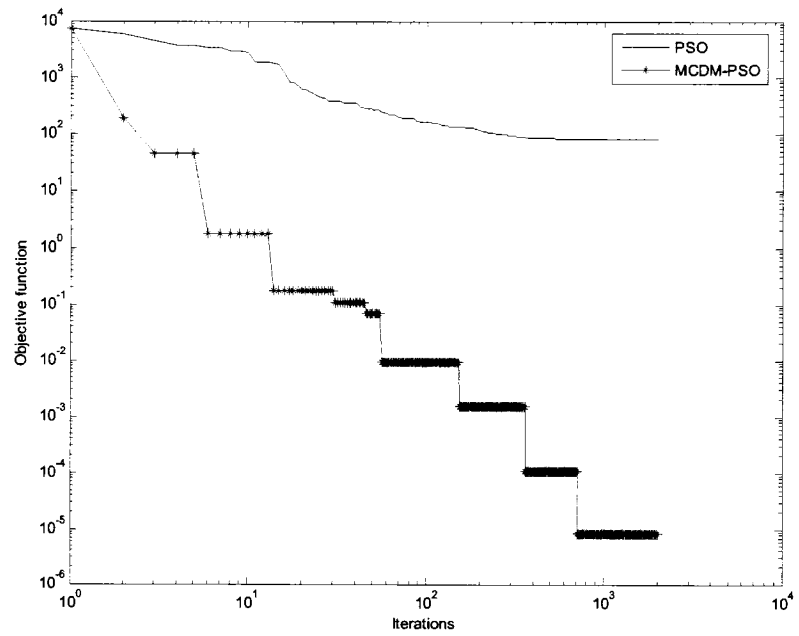


Figure 6.25: Convergence of the 30 dimensions Rosenbrock function.



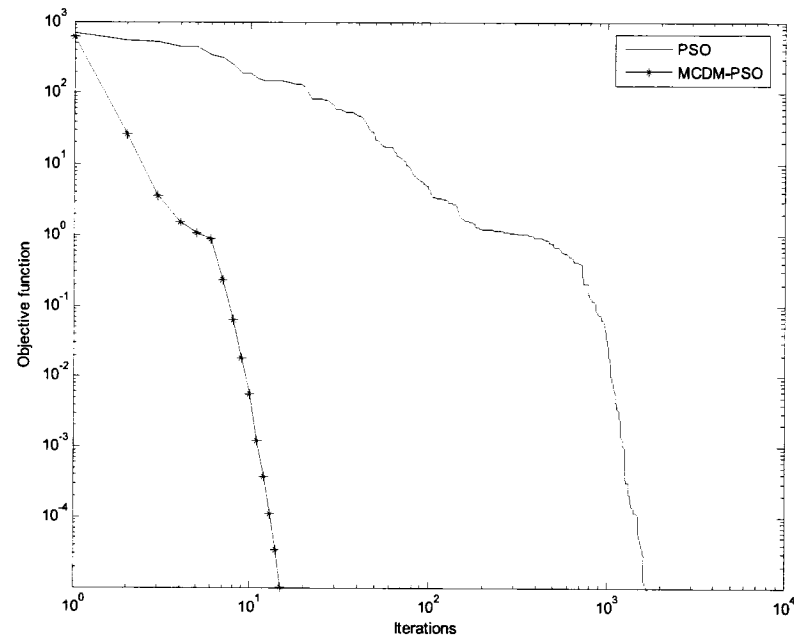


Figure 6.26: Convergence of the 30 dimensions Griewangk function.

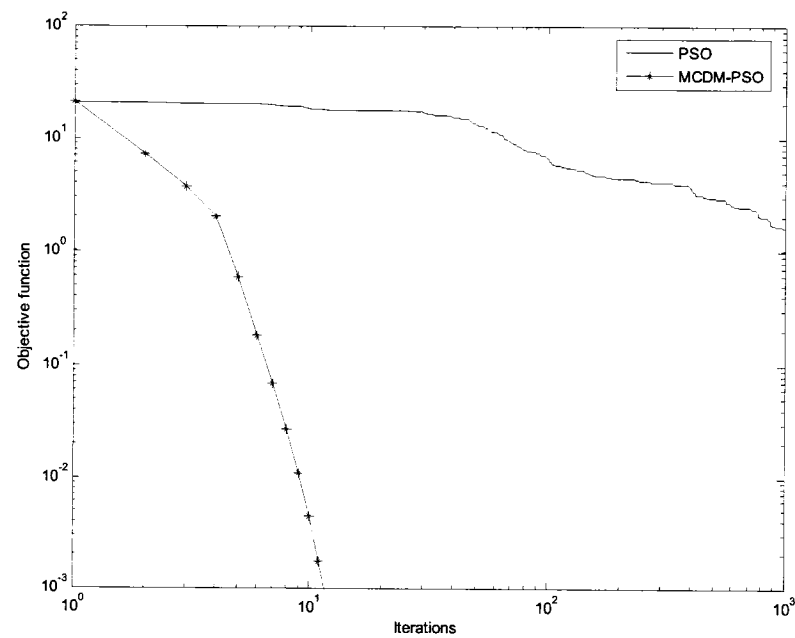
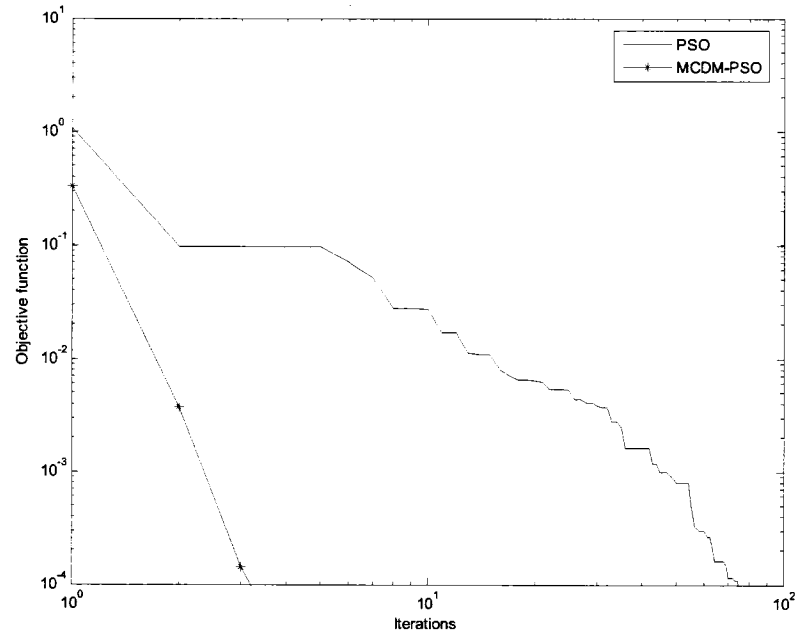
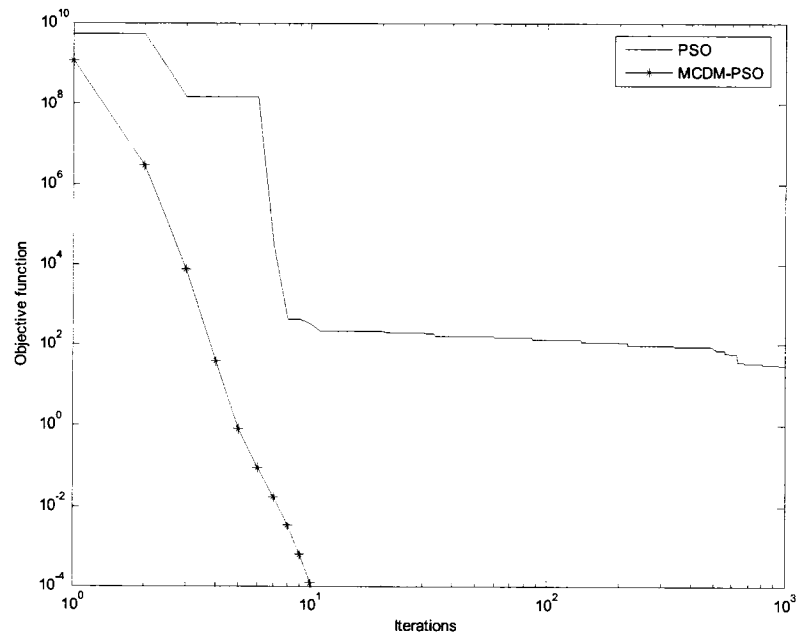


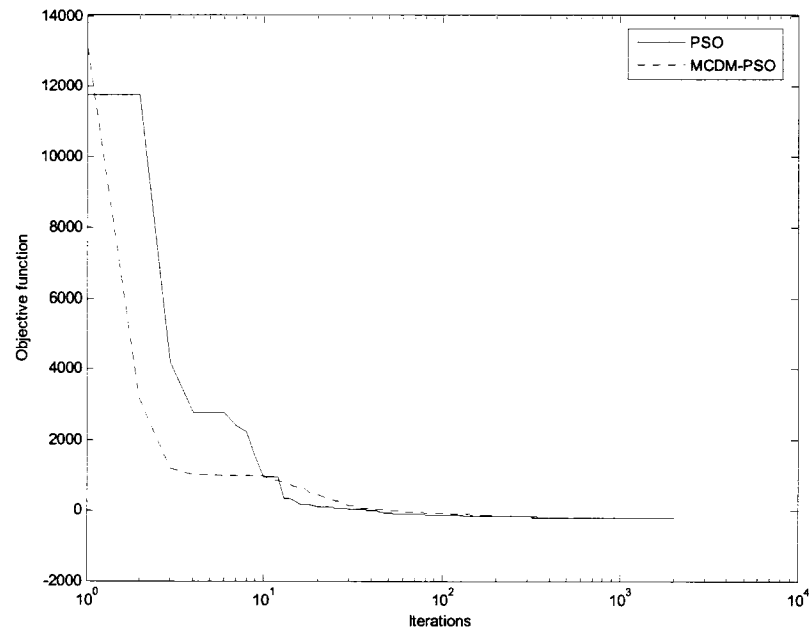
Figure 6.27: Convergence of the 30 dimensions Ackley function



**Figure 6.28: Convergence of the 30 dimensions The sum of different powers function.**



**Figure 6.29: Convergence of the 30 dimensions Zakharov function.**



**Figure 6.30: Convergence of the 10 dimensions Trid function.**

#### 6.4.2 MCDM-PSO vs. other algorithms

First, the proposed MCDM-PSO is compared with the attractive-repulsive particle swarm optimizer (ARPSO) and the simple evolutionary algorithm (SEA). Results for ARPSO and SEA are obtained by the authors of reference [145]. According to the authors, the maximum number of evolutions allowed for ARPSO and SEA was set to 500,000 for problems of 30 dimensions or less and 5,000,000 for problems of 100 dimensions. The author used a swarm size of 25 particles for the ARPSO and a population size of 100 for the SEA. His results were averaged over 30 runs for each problem. To be consistent, the results are also averaged over 30 runs for all problems used in the comparison. Results obtained by the proposed MCDM-PSO, ARPSO and SEA in cases of 30 or less dimensions and 100 dimensions are listed in Tables 6.7 and 6.8 respectively. In the tables, the entry BKS describes the best known solution of each problem. Also, the number 0.0000000e+00 means that the solution is less than  $1e-25$ .

From Table 6.7, it is clear that results obtained by the MCDM-PSO for problems of 30 or less dimensions are far better than those of the ARPSO especially with  $f_1, f_4, f_5, f_6, f_7, f_8$ , and  $f_{19}$  where differences are very significant. Also the proposed MCDM-PSO outperforms the SEA in all cases except for  $f_{14}$  and  $f_{15}$ . However, in these two specific problems differences on solutions obtained by the MCDM-PSO and the SEA are insignificant.

For problems with 100 dimensions, results are presented in the Table 6.8. In all test problems of this group, the proposed MCDM-PSO performs significantly better than both the ARPSO and the SEA. Also variations of the results over the 30 runs for the MCDM-PSO are very small compared to those of the ARPSO and the SEA.

Table 6.7: Results for MCDM-PSO, ARPSO and SEA on problems of 30 dimensions or less.

Problem No.	BKS	MCDM-PSO		ARPSO		SEA	
		Mean	Std.	Mean	Std.	Mean	Std.
1	0	<b>0.0000000e+00</b>	0.000e+00	6.8081735e-13	5.3e-13	1.7894112e-03	2.77e-04
2	0	<b>0.0000000e+00</b>	0.000e+00	<b>0.0000000e+00</b>	2.13e-25	1.5891817e-02	4.25e-03
4	0	<b>0.0000000e+00</b>	0.000e+00	2.1491414	4.91	7.1789575e-01	9.22e-01
5	-1.2569e+4	<b>-1.2569443e+4</b>	0.06757	-8.5986527e+3	2.07e+03	-1.1669334e+4	2.34e+02
6	0	<b>6.3956471e-5</b>	1.00e-4	3.5509286e+2	2.15e+03	3.1318954e+01	1.74e+01
7	0	<b>1.9613940e-16</b>	2.74e-16	9.2344555e-2	3.41e-01	4.6366988e-03	3.96e-03
8	0	<b>8.8817841e-16</b>	0.000e+00	1.8422773e-7	7.15e-08	1.0468180e-02	9.08e-04
14	-1.0316	-1.031588	2.64e-5	-1.0316284	3.84e-08	<b>-1.031630</b>	3.16e-08
15	0.3978	0.3979103	4.34e-5	0.39788736	5.01e-09	<b>0.39788700</b>	2.20e-08
19	-10.5	-10.53637	5.84e-5	-8.6155040	2.88	-9.7995696	2.24e+00

**Table 6.8: Results for MCDM-PSO, ARPSO and SEA on problems of 100 dimensions.**

Problem No.	BKS	MCDM-PSO		ARPSO		SEA	
		Mean	Std.	Mean	Std.	Mean	Std.
1	0	0.0000000e+00	0.000e+00	7.4869991e+02	2.31e+03	5.2291447e-04	5.18e-05
2	0	0.0000000e+00	0.000e+00	1.8174752e+01	2.50e+01	3.6846433e-02	6.06e-03
4	0	0.0000000e+00	0.000e+00	4.8096522e+01	9.54e+00	9.9767318e-02	3.04e-01
5	-4.189e4	-4.1898185e+4	0.2628846	-2.1209102e+4	2.98e+03	-3.9430820e+4	5.36e+02
6	0	1.6732261e-4	2.15e-4	2.3609401e+02	1.25e+02	9.2492247e+01	1.29e+01
7	0	7.0314124e-17	1.21e-16	8.5311042e-02	2.56e-01	1.8932321e-03	4.42e-03
8	0	8.8817841e-16	0.000e+00	5.6281044e-02	3.08e-01	2.9328603e-03	1.47e-04

Second, the proposed algorithm is compared to five other important PSO variants as described above. These variants are accompanied with different schemes to overcome the problem of premature convergence and improve PSO performance in general. The comparison includes problems  $f_1, f_4, f_5, f_6, f_7$ , and  $f_8$  on 30 dimensions. Results for PSO variants used in the comparison are originally presented in reference [138]. According to the authors, for all PSO variants, a swarm size of 40 is used and a maximum number of function evolutions is set to 200,000. Also results are based on an average of 30 runs for each problem. Table 6.9 shows that the proposed MCDM-PSO performs better than any other PSO variant for all problems except for  $f_8$  where a comprehensive learning particle swarm optimizer CLPSO shows a slightly better value.

**Table 6.9: MCDM-PSO vs. some PSO variants on problems of 30 dimensions.**

Problem No.	BKS	MCDM-PSO	UPSO	FDR-PSO	FIPS	CPSO-H	CLPSO
		Mean	Mean	Mean	Mean	Mean	Mean
1	0	0.0000000e+00	0.000e+00	0.000e+00	2.69e-13	0.000e+00	4.46e-14
4	0	0.0000000e+00	6.59e+1	2.84e+1	7.3e+1	0.000e+00	4.85e-10
5	-1.2569e+4	-1.2569e+4	-7.729e+3	-8.959e+3	-1.051e+4	-1.148e+4	-1.2569e+4
6	0	6.3956471e-5	1.51e+1	5.39e00	2.45e+1	7.08e00	2.1e+1
7	0	1.9613940e-16	1.66e-3	1.01e-2	1.16e-6	3.63e-2	3.14e-10
8	0	8.8817841e-16	1.22e-15	2.84e-14	4.81e-7	4.93e-14	0.000e+00

### 6.5 Summary

In this chapter, the proposed MCDM-PSO was extended to handle the continuous optimization problems. A set of 32 diverse problems was chosen from the literature to test the performance of the proposed algorithm. Based on the obtained results and the comparison with the traditional PSO, one can find that the proposed technique has a better performance and a greater resistance to be trapped in local optima. Also, the performance of the MCDM-PSO was compared to other PSO variants. Results of the comparison showed superior performance over these variants for almost all the test benchmark problems.

## **Chapter 7: Conclusions and future work**

### **7.1 Summary**

This thesis presents a new swarm optimization technique. Unlike the traditional particle swarm optimizer PSO, the proposed technique is not based on mere observation of birds flocking. Instead, a well-established theory in the field of sociobiology is manipulated to form the swarm and keep its persistence. Specifically, gregarious and social intolerance behaviors demonstrated by social animals are adopted to introduce another set of lower level behaviors that are easy to manipulate. This set of basic behaviors includes imitation, memory retrieval, play, and momentum. The link between this set of behaviors and gregariousness and social intolerance, along with the proof of existence of such behaviors in the social animal world are introduced in chapter 3. Agents of the proposed swarm are able to respond adaptively to changes made by their swarm mates by controlling the proposed set of basic behaviors. This gives the proposed swarm a more natural look by adding the aspect of freewill to each agent in the swarm. In order



for the agents to determine how to respond to changes of their flock mates, a multi-criterion decision making process is employed instead of the simple additive weighted logic used to update particles moves in the PSO. Moreover, the distance factor is directly added to the position update equation in addition to fitness to reduce the chance of premature convergence that could take place due to the lack of diversity. Therefore, decisions made by agents are to promote both fitness and diversity. Among the new mechanisms that are applied in this thesis is the Levenshtein edit distance. Positions of agents in the domain space are expressed by permutations. Distances between agents in the genotype space are measured by how similar are their permutations. Agents respond negatively to those who are very close (social intolerance) and vice versa. Also, decisions are made to favor those who have high fitness in the phenotype space. In coding language, negative and positive responses are meant to control the number of items transmitted by an agent to its permutation from the permutations representing others who are involved in the decision making process depending on a decision index. To reach a meaningful decision index, the fuzzy ordered weighted average (OWA) is used to aggregate the diversity and the fitness criteria in a rational manner. Responses and changes of each agent's state within the swarm are elevated to help the system to iterate longer and prevent stagnation.

The proposed technique is first tested with a few instances of the traveling salesmen problems (TSP) from both symmetrical and asymmetrical types. The results are very promising and agree to high extent to the best known solutions in the literature. The technique is able to find near optimal solutions by testing few fractions of all possible tours.

Afterwards, the technique is extensively tested with the quadratic assignment problem (QAP). The QAP is a very important model in the field of combinatorial optimization due to its generality and ability to fit many assignment problems in the real world. Despite its importance, the QAP seems not to be adequately addressed by the swarm based techniques in the literature. In this work, 131 QAP problems with different sizes and structures are tested using the proposed technique. The results are compared with the results reached by the standard genetic algorithm (SGA) in terms of average and best gaps between the best known solutions and solutions obtained over 10 runs. To test

the significance of the difference between the error samples achieved by both techniques, the nonparametric Mann-Whitney test of significance is applied for both average and best results. The results of the test show that the errors in the case of the SGA are significantly higher than those of the proposed technique for both average and best cases and there is no chance that both error samples would be drawn from the same population.

The performance of the proposed technique is compared with the ant colony optimizer (ACO) and the traditional particle swarm optimizer (PSO) on some QAP problems. Results are averaged over ten replications and the mean and the standard deviations of the obtained results are recorded. For all the studied problems in this case, the proposed technique shows superior performance over the both techniques in terms of the average results. However, higher standard deviations are observed with some problems due to the use of small swarm size (five agents).

The technique is also extended to solve the continuous optimization problems where variables are represented by real values. In this case, the Levenshtein edit distance is eliminated and the distances between agents are expressed by the simple differences between their variables. The position update equation in this case is set as a weighted difference equation where weights are the decision indices described earlier. The technique is tested with a large, diverse set of benchmark problems. A total of 32 problems with different dimensions ranging from 2 to 100 are used. The test problems also include unimodal and multimodal types. The multimodal types are specifically chosen to test the ability of the proposed technique to resist trapping in local optima.

The performance of the proposed technique is compared with the performance of the traditional PSO. Solutions are recorded in terms of the means and the standard deviations over 30 runs for the problems with dimensions of 30 or higher and over 20 runs for those with less than 30 dimensions. The results obtained by the proposed technique for problems with 30 dimensions or higher are far better than the results obtained by the PSO. Also the iterations vs. objective function plots for the studied problems show faster convergence in the case of the proposed technique. For problems with dimensions less than 30, both techniques perform equally well.

The performance of the technique is also compared with six PSO variants that were proposed to overcome the problem of the premature convergence in PSO. The

results obtained by the proposed technique in almost all the studied problems are better than those achieved by any of these variants.

## **7.2 Future work**

Generally, the proposed technique shows very promising performance with both combinatorial and continuous optimization problems. However, there are many issues that need to be explored. For example, the significance of the ordered weights values and the effect of the decision strategies on the performance of the technique need to be studied. The influence of the neighborhood topology on the performance of the agents inside the swarm is also another subject that could be studied in the future. The performance of the proposed technique with the constrained optimization problems can also be one of the directions for future studies.

## Bibliography

1. Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics", Proceedings of SIGGRAPH Conference, Vol. 21, No. 4, pp. 25-34, 1987.
2. O'Sullivan, D. and Haklay, M., "Agent-based models and individualism: is the world agent-based?", Journal of Environment and Planning A, Vol. 32, No. 8, pp. 1409-25, 2000.
3. Kennedy, J. and Eberhart, R., "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, Piscataway, NJ, 1995.
4. Villiers, D., "Using Particle Swarm Optimization for Offline Training in a Racing Game", Game Developer Magazine, 27 February, 2006. Available at: [http://www.gamasutra.com/features/20051213/villiers\\_01.shtml](http://www.gamasutra.com/features/20051213/villiers_01.shtml).
5. Asproth, V., Holmberg, S. and Hakansson, A., "Decision Support functions embedded in GIS," in The Nordic GIS Conference, Helsinki, October, 2001.
6. Iztok, L., Miha, M. and Nikolaj, Z., "Boids with a fuzzy way of thinking", Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, Vol.7; pp.58-62, 2003.
7. Mendes, R., *Population Topologies and Their Influence in Particle Swarm Performance*, PhD thesis, Departamento de Inform'atica, Escola de Engenharia, Universidade do Minho, 2004.
8. Xie, X., Zhang, W. and Yang Z., "A dissipative particle swarm optimization", Proceedings of the Fourth Congress on Evolutionary Computation (CEC), Vol. 2, pp 1456-1461, Hawaii, USA, 2002.
9. Løvbjerg, M. and Krink, T., "Extending Particle Swarms with Self-Organized Criticality", Proceedings of the Fourth Congress on Evolutionary Computation (CEC), Vol. 2, pp. 1588-1593, Hawaii, USA, 2002.
10. Krink, T., Vesterstrøm, S. and Riget, J., "Particle Swarm Optimization with Spatial Particle Extension", Proceedings of the Fourth Congress on Evolutionary Computation (CEC), Vol. 2, pp. 1474-1479, Hawaii, USA, 2002.

11. Blackwell, T. and Bentley, P., "Don't push me! Collision avoiding swarms", Proceedings of the Fourth Congress on Evolutionary Computation (CEC), Vol. 2, pp. 1991-1696, Hawaii, USA, 2002.
12. Parsopoulos, K., Plagianakos, V., Magoulas, G. and Vrahatis, M., "Stretching technique for obtaining global minimizes through particle swarm optimization", Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN, 2001.
13. Eberhart, R., Yuhui, S. and Kennedy, J., *Swarm Intelligence*, The Morgan Kaufmann Series in Artificial Intelligence 2001.
14. Pulido, G. and Coello, C., "A constraint-handling mechanism for particle swarm optimization", Proceedings of Evolutionary Computation (CEC) Congress, Vol. 2, pp. 1396-1403, June, 2004.
15. Ho, S., Yang, S., Guangzheng, N., Lo, E. and Wong H., "A particle swarm optimization-based method for multiobjective design optimizations", IEEE on Magnetics, Vol. 41, Issue 5, pp. 1756-1959, May, 2005.
16. Belapaeme, T., Boer, B., Vylder, B. and Jansen B., "The role of population dynamics in imitation", Proceedings of the 2<sup>nd</sup> International Symposium on Imitation in animals and Artercraft, pp. 171-175, 2003.
17. Franklin, S. and Graesser, A., "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," In Intelligent Agents III, Lecture Notes on Artificial Intelligence, pp. 21--35. Springer-Verlag, Berlin, 1997.
18. The Levenshtein edit distance, Available online at [http://en.wikipedia.org/wiki/Levenshtein\\_distance](http://en.wikipedia.org/wiki/Levenshtein_distance).
19. Logsdon, A., "Abstract Reasoning", Your Guide to Learning Disabilities. <http://learningdisabilities.about.com/od/glossar1/g/abstractreason.htm>
20. Cooper, N., "An historical perspective: from Turing and von Neumann to the present", Journal of Los Alamos Science, No. 9, pp. 22-27, 1983.
21. Tempesti, G., Mange, D., and Stauffer, A., "Self-replicating and Self-Repairing Multi-cellular Automata", Artificial Life, Vol. 4, No. 3, pp. 259-282, 1998.
22. Marchal, P., "John von Neumann: the founding father of artificial life", Artificial Life Vol. 4, No. 3, pp. 229-235, 1998.

23. Sipper, M., *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Springer-Verlag, Heidelberg, 1997.
24. Gage, D. et al. "Cellular automata: is rule 30 random?", Midwest NKS Conference, Bloomington, IN, October, 2005.
25. Wolfram, S., *A New Kind of Science*, Wolfram Media Publisher, Champaign, IL, 2002.
26. Weisstein, E., "Cellular Automaton", From *MathWorld*--A Wolfram Web Resource. <http://mathworld.wolfram.com/CellularAutomaton.html>.
27. Anderson, P., "More Is Different", *Science*, New Series, Vol. 177, No. 4047, pp. 393-396, 1972.
28. Fromm, J., *The Emergence of Complexity*, Kassel University press GmbH, Kassel, Germany, 2004.
29. Shi, Y. and Eberhart, R., "A modified particle swarm optimizer", Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 69-73. Piscataway, NJ, 1998.
30. Shi, Y. and Eberhart, R., "Parameter selection in particle swarm optimization", Proceedings of the 7th International Conference on Evolutionary Programming VII, LNCS Vol. 1447, pp. 591-1600, New York, 1998.
31. Zheng, Y., Ma, L., Zhang, L. and Qian, J., "Empirical study of particle swarm optimizer with an increasing inertia weight", Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 221-226, Piscataway, NJ, 2003.
32. Zheng, Y., Ma, L., Zhang, L. and Qian, J., "On the convergence analysis and parameter selection in particle swarm optimization", Proceedings of the Second International Conference on Machine Learning and Cybernetics, Vol. 3, pp. 1802- 1807, 2003.
33. Wolpert, D. and Macready, W., "No Free Lunch Theorems for Optimization", IEEE Transaction on Evolutionary Computation, Vol. 1, No. 1, pp 67-82, April, 1997.
34. Chatterjee, A. and Siarry, P., "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization", Computers & Operations Research, Vol. 33, pp. 859-871, 2006.

35. Lei, K., Qiu, Y. and He, Y., "A New Adaptive Well-Chosen Inertia Weight Strategy to Automatically Harmonize Global and Local Search Ability in Particle Swarm Optimization", Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics, (ISSCAA 2006), pp. 977-980, 2006.
36. Chen, G. et al., "Self-Active Inertia Weight Strategy in Particle Swarm Optimization Algorithm", Proceedings of the 6th World Congress on Intelligent Control and Automation, pp. 3686-3689, 2006.
37. Chen, G. et al., "Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization", Proceedings of the 6th World Congress on Intelligent Control and Automation, pp. 3672-3675, 2006.
38. Clerc, M. and Kennedy, J., "The particle swarm - explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation Vol. 6, No. 1, pp. 58 – 73, February, 2002.
39. Ozcan, E. and Mohan, C.K., "Analysis of a simple particle swarm optimization system", Proceedings of Intelligent Engineering Systems through Artificial Neural Networks, Vol. 8, pp. 253–258, St Louis, Missouri, November, 1998.
40. Ozcan, E. and Mohan, C.K., "Particle swarm optimization: surfing the waves", Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999), Vol. 3, pp. 1939–1944, Washington, DC, USA, 1999.
41. Van den Bergh, F., *An Analysis of Particle Swarm Optimizers*, PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
42. Jiang, M., Luo, Y. P. and Yang, S. Y., "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm", Information Processing Letters, Vol. 102, Issue 1, pp. 8-16, April, 2007.
43. Kadiramanathan, V., Selvarajah, K. and Fleming, P., "Stability analysis of the particle dynamics in particle swarm optimizer", IEEE Transactions on Evolutionary Computation, Vol. 10, No. 3, June, 2006.

44. Zhihua, C. and Jianchao, Z., "A Guaranteed Global Convergence Particle Swarm Optimizer", *Rough Sets and Current Trends in Computing*, Springer-Verlag Publisher, Vol. 3066, pp. 762-767, 2004.
45. Wang, R. and Zhang, X., "Particle Swarm Optimization with Opposite Particles", *Proceedings of the 4th Mexican International Conference on Artificial Intelligence*, pp. 633-640, Monterrey, Mexico, November, 2005.
46. Solis, F. and Wets, R., "Minimization by random search techniques", *Mathematics of Operation Research*, Vol. 6, pp. 19-30, 1981.
47. Van den Bergh, F. and Engelbrecht A., "A new locally convergent particle swarm optimizer", *Proceedings of the IEEE International Conference on Man, System and Cybernetics*, Vol. 3, Tunisia, 2002.
48. Heppner, F. and Grenander, U., "A stochastic nonlinear model for coordinated bird flocks", in: Krasner, S., (Ed.), *The Ubiquity of Chaos*, AAAS publications, Washington, pp. 233-238, 1990.
49. Riget, J. and Vesterstrom, J., "A diversity-guided particle swarm optimizer - the ARPSO", *Technical Report 2002-02*, Department of Computer Science, University of Aarhus, Denmark, 2002.
50. Peram, T., Veeramachaneni, K. and Mohan, C., "Fitness-distance-ratio based particle swarm optimization", *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 174-181, 2003.
51. Blackwell, T. and Bentley, P., "Dynamic search with charged swarms", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 19-26, 2002.
52. Blackwell, T., "Particle swarms and population diversity", *Journal of Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Vol. 9, No. 11, pp. 793-802, November, 2005.
53. Yen, G. and Daneshyari, M., "Diversity-based Information Exchange among Multiple Swarms in Particle Swarm Optimization", *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 1686-1693, July, 2006.



54. El-Abd, M. and Kamel, M., "Information exchange in multiple cooperating swarms", Proceedings of the IEEE Swarm Intelligence Symposium (SIS), pp. 138-142, June, 2005.
55. Baskar, S. and Suganthan, P., "A novel concurrent particle swarm optimization", Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Vol. 1, pp. 792-796, June, 2004.
56. Jones, W., "Complex Adaptive Systems," Beyond Intractability. Eds. Guy Burgess and Heidi Burgess, Conflict Research Consortium, University of Colorado, Boulder, Oct. 2003. Available online at:  
[http://www.beyondintractability.org/essay/complex\\_adaptive\\_systems/](http://www.beyondintractability.org/essay/complex_adaptive_systems/)
57. John, E., "Flocking Behavior in Birds," The Auk Journal, Vol. 69, pp. 160-170, April, 1952.
58. Craig, W., "Appetites and Aversions as Constituents of Instincts," Proceedings of the National Academy of Sciences of the United States of America, Vol. 3, No. 12, pp. 685-688, December, 1917.
59. Emlen, J. T. and Lorenz, F. W., "Pairing Responses of Free-living Valley Quail to Sex Hormone pellet implants." The Auk Journal, vol. 59, pp. 509-527, July, 1942.
60. Collias, N., "Statistical Analysis of Factors Which Make for Success in Initial Encounters between Hens", The American Naturalist, Vol. 77, No. 773, pp. 519-538, 1943.
61. Scaruffi, P., Thinking about Thought: A Primer on the New Science of Mind, Towards a Unified Understanding of Mind, Life and Matter. Writers Club Press, 2003,
62. Milius, S., "Where'd I Put That? Maybe it takes a bird brain to find the car keys," Science News Magazine, vol. 165, February 14, 2004. Available online at:  
<http://www.sciencenews.org/articles/20040214/bob8.asp>
63. Owen, J., "Memory Aids Birds in Migration, Study Finds," National Geographic News, April 29, 2003. Available online at:  
[http://news.nationalgeographic.com/news/2003/04/0429\\_030429\\_birdbrains.html](http://news.nationalgeographic.com/news/2003/04/0429_030429_birdbrains.html)

64. Milius, S., "Birds with a Criminal Past Hide Food Well," Science News Magazine, Vol. 160, November, 2001. Available online at:  
[http://findarticles.com/p/articles/mi\\_m1200/is\\_21\\_160/ai\\_81110067](http://findarticles.com/p/articles/mi_m1200/is_21_160/ai_81110067)
65. Milius, S., "Birds can Remember What, Where, and When," Science News Magazine, Vol. 154, September, 1998. Available online at:  
<http://www.sciencenews.org/pages/pdfs/data/1998/154-12/15412-04.pdf>
66. Suddendorf, T. and Busby, J., "Mental Time Travel in Animals?" TRENDS in Cognitive Sciences, Vol. 7, pp. 391-396, September, 2003.
67. Clayton, N., Bussey, T. and Dickinson, A., "Can Animals Recall the Past and Plan for the Future?" Nature Reviews. Neuroscience, Vol. 4, pp. 685-691, August, 2003.
68. Suddendorf, T. and Busby, J., "Like it or Not? The Mental Time Travel Debate: Reply to Clayton et al." TRENDS in Cognitive Sciences, Vol. 7, pp. 437-438, October, 2003.
69. Clayton, N. and Dickinson, A., "Episodic-like memory during Cache Recovery by Scrub Jays," The Nature Journal, Vol. 395, pp. 272-274, September, 1998.
70. Galef, B. and Laland, K., "Social learning in Animals: Empirical Studies and Theoretical Models", BioScience, Vol.55 No.6, pp. 489-499, 2005.
71. Noble, J. and Todd, P., "Is it really imitation? A review of simple mechanisms in social information gathering," in Proceedings of the AISB'99 Symposium on Imitation in Animals and Artifacts, pp. 65-73, 1999.
72. Zentall, T. and Akins, C., "Imitation in animals: evidence, function and mechanisms", In Robert Cook (Eds.), Avian Visual Cognition, Comparative Cognition Press, 2001.  
 Available online at: <http://www.pigeon.psy.tufts.edu/avc/zentall/default.htm>.
73. Alonso, E. et al, "Learning in multi-agent systems", The Knowledge Engineering Review, Vol. 16, No.3, pp. 277-284, Cambridge University Press, 2001.

74. Yager, R., "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision making," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, pp. 183-190, January, 1988.
75. Fulop, J., "Introduction to decision making methods," *Biodiversity & Ecosystem Informatics Workshop (BDEI-3)*, The Evergreen State College, Olympia, Washington, December 13–15, 2004.
76. Eastman, J., "Decision Strategies in GIS," *Directions Magazine*, Dec. 2000, Available online at [www.Directionsmag.com](http://www.Directionsmag.com).
77. Husdal, J., "Geographical Decision Making - Different approaches in IDRISI", Available online at <http://www.husdal.com/mscgis/gdm.htm>, 1999.
78. Gilleland, M., "Levenshtein Distance, in Three Flavors," Available online at: [www.merriampark.com/ld.htm](http://www.merriampark.com/ld.htm).
79. Mendes, J. and Motizuki, W., "Urban Quality of Life Evaluation Scenarios: The Case of Sao Carlos in Brazil," *The professional Journal of the Council on Tall Buildings and Urban Habitat (CTBUH Review)*, Vol. 1, pp. 1-11, February, 2001.
80. Larsen, H., "Importance weighted OWA aggregation of multi-criteria queries", *Proceeding of the North American Fuzzy Information Processing Society Conference*, pp. 740-744, June, 1999.
81. Rinner, C. and Raubai, M., "Personalized Multi-Criteria Decision Strategies in Location-based Decision Support", *Journal of Geographic Information Sciences*, Vol. 11, pp. 149-156, June, 2005.
82. Larranaga, P., et al., "Genetic algorithms for the traveling salesman problem: A review of representations and operators", *Artificial Intelligence Review*, Vol. 13, pp. 129-170, 1999.
83. Schrijver, A., "On the history of combinatorial optimization (till 1960)", in: *"Handbook of Discrete Optimization"* (K. Aardal, G.L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, pp. 1-68, 2005.

84. Held, M., Hoffman, A., Johnson, E. and Wolfe, P., "Aspects of the traveling salesman problem", IBM journal of Research and Development, Vol. 28, NO. 4, pp. 476-486, July, 1984.
85. Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., "Solution of a Large-scale Traveling Salesman Problem", Journal of Operations Research, Vol. 2, pp. 393-410, October, 1954.
86. Malkevitch, J., Sales and Chips, feature column on mathematical topics AMS  
<http://www.ams.org/featurecolumn/archive/tsp.html>
87. Laporte, G. and Osman, I. H., "Routing problems: A bibliography", Annals of Operations Research, Vol. 61, pp.227–262, 1995.
88. Petchenkine, V. , software package for graphs [www.geocities.com/pechv\\_ru/](http://www.geocities.com/pechv_ru/)
89. Weisstein, E. W., "Floor Function", From *MathWorld*--A Wolfram Web Resource.  
<http://mathworld.wolfram.com/FloorFunction.html>
90. Reinelt, G., "TSPLIB - a Traveling Salesman Problem Library", Journal of Operation Research Society of America on Computing (ORSA), Vol. 3, No. 4, pp. 376-384, 1991.
91. Nugent, C.E., Vollman, T.E., and Ruml, J., "An experimental comparison of techniques for the assignment of facilities to locations" Journal of Operations Research, Vol. 16, No. , pp. 150–173, 1968.
92. Elshafei, A.N., "Hospital layout as a quadratic assignment problem", Operations Research Quarterly Vol. 28, No. 1, Part 2, pp. 167–179, 1977.
93. Steinberg, L., "The backboard wiring problem: a placement algorithm" Journal of the Society of Industrial and Applied Mathematics (SIAM Review), Vol. 3, No. 1, pp. 37–50, 1961.
94. Burkard, R., Karisch, S. and Rendl, F., "QAPLIB – A Quadratic Assignment Problem Library", Journal of Global Optimization, Vol. 10, pp. 391–403, 1997.
95. Eschermann, B. and Wunderlich, H., Optimized synthesis of self-testable finite state machines. Proceedings of the 20th International Symposium on Fault-Tolerant Computing (FFTCS 20), Newcastle upon Tyne, pp. 390-397, June, 1990.

96. Christofides, N. and Benavent, E., "An exact algorithm for the quadratic assignment problem", *Journal of Operations Research*, Vol. 37, No.5, pp. 760–768, 1989.
97. Hadley, S.W., Rendl, F., and Wolkowicz, H., "A new lower bound via projection for the quadratic assignment problem", *Journal of Mathematics of Operations Research*, Vol. 17, No.3, pp. 727–739, 1992.
98. Scriabin, M. and Vergin, R.C., "Comparison of computer algorithms and visual based methods for plant layout", *Journal of Management Science*, Vol. 22, No.2, pp. 172–187, 1975.
99. Lashkari, R. and Jaisingh, S., "A Heuristic Approach to Quadratic Assignment Problem", *Journal of the Operational Research Society*, Vol. 31, No. 9., pp. 845–850, 1980.
100. Hillier, F. and Connors, M., "Quadratic assignment problem algorithms and the location of indivisible facilities", *Journal of Management Science*, Vol. 13, No. 1, pp. 42–57, 1966.
101. Li, Y. and Pardalos, P.M., "Generating quadratic assignment test problems with known optimal permutations" *Journal of Computational Optimization and Applications*, Vol. 1, No. 2, pp. 163–184, 1992.
102. Pardalos, P., "Generation of large-scale quadratic programs for use as global optimization test problems", *ACM Transactions on Mathematical Software*, Vol. 13, No. 2, pp. 133–137, 1987.
103. Knowles, J. and Corne, D., "Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem", *Proceedings of Evolutionary Multi-Criterion Optimization (EMO 2003) Second International Conference*, Faro, Portugal, pp. 295–310, April, 2003.
104. Romeijn, H. and Morales, D., "Generating Experimental Data for the Generalized Assignment Problem", *Journal of Operations Research*, Vol. 49, No. 6, pp. 866–878, 2001.
105. Taillard, E.D., "Comparison of iterative searches for the quadratic assignment problem", *Journal of Location Science*, Vol. 3, No. 2, pp. 87–105, August, 1995.

106. Anstreicher, K. et al., "Nug30 is solved!!", Available online at <http://www-unix.mcs.anl.gov/metaneos/nug30/>.
107. Land, A. H., Doig, A. G. , "An Automatic Method of Solving Discrete Programming problems" *Journal of the Econometric society (Econometrica)*, Vol. 28, No. 3, pp. 497-520, July, 1960.
108. Anstreicher, K., Brixius, N., Goux, J. and Linderoth, J., "Solving Large Quadratic Assignment Problems on Computational Grids", *Journal of Mathematical Programming*, Vol. 91, No. 3, pp. 563-588, 2002.
109. Mavridou, T. and Pardalos, P., "Simulated Annealing and Genetic Algorithms for the Facility Layout Problem: A Survey", *Journal of Computational Optimization and Applications*, Vol. 7, pp. 111-126, 1997.
110. Tian, P. Wang, H. and Zhang, D., "Simulated annealing for the quadratic assignment problem: A further study *Proceedings of the 18th International Conference on Computers and Industrial Engineering*, Vol. 31, No. 3-4, pp. 925-928, December, 1996.
111. Nissen, V. and Paul, H., "A modification of threshold accepting and its application to the quadratic assignment problem", *Operation Research (OR) Spectrum*, Vol. 17, pp. 205–210, 1995.
112. Yip, P. and Yoh-Han, P., "Guided evolutionary simulated annealing approach to the quadratic assignment problem", *IEEE transactions on systems, man, and cybernetics*, Vol. 24, No. 9, pp. 1383-1386, 1994.
113. Connolly, D., "General Purpose Simulated Annealing", *The Journal of the Operational Research Society*, Vol. 43, No. 5, pp. 495-505, 1992.
114. Skorin-Kapov, J., Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2:33–45, 1990.
115. Chakrapani, J. and Skorin-Kapov, J., "Massively parallel tabu search for the quadratic assignment problem", *Annals of Operations Research*, Vol. 41, pp.327–341, 1993.
116. Punnen, A. and Aneja, Y., "A Tabu Search Algorithm for the Resource-Constrained Assignment Problem", *The Journal of the Operational Research Society*, Vol. 46, No. 2., pp. 214-220, 1995.

117. Taillard, E.D., Robust taboo search for the quadratic assignment problem. *Parallel Computing*, Vol. 17, pp. 443–455, 1991.
118. Battiti, R. and Tecchiolli, G., “The reactive tabu search”, *Journal of Operation Research Society of America on Computing (ORSA)*, Vol. 6, No. 2, pp. 126–140, 1994.
119. Maniezzo, V., Dorigo, M. and Colomi, A., The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Universit’e Libre de Bruxelles, Belgium, 1994.
120. Gambardella, L.M., Taillard, E.D. and Dorigo, M., “Ant colonies for the quadratic assignment problem”, *Journal of the Operational Research Society*, Vol. 50, No. 2, pp. 167–176, 1999.
121. Maniezzo, V., “Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem”, *INFORMS Journal on Computing*, Vol. 11, No. 4, pp. 358–369, 1999.
122. Maniezzo, V. and Colomi, A., “The Ant System applied to the quadratic assignment problem”, *IEEE Transactions on Data and Knowledge Engineering*, Vol. 11, No. 5, pp.769–778, 1999.
123. Chu, P. and Beasley, J., “A Genetic Algorithm for the Generalized Assignment Problem”, *Journal of Computers and Operations Research*, Vol. 24, No. 1, pp.17–23, 1997.
124. Misevicius, A., “An improved hybrid genetic algorithm: new results for the quadratic assignment problem”, *Journal of Knowledge-based systems*, Vol. 17, No.2, pp. 65-73, 2004.
125. Drezner, Z., “A new genetic algorithm for the quadratic assignment problem”, *INFORMS Journal on Computing*, Vol. 15, No. 3, pp. 320–330, 2003.
126. Hanan, M., and Kurtzberg, J., “A Review of the Placement and Quadratic Assignment Problems” *Journal of the Society of Industrial and Applied Mathematics (SIAM Review)*, Vol. 14, No. 2, pp. 324-342, April, 1972.
127. Koopmans, T.C. and Beckmann, M.J., “Assignment problems and the location of economic activities,” *Journal of the Econometric society (Econometrica)*, Vol. 25, No. 1, pp. 53–76, 1957.

128. Stützle, T. and Dorigo, M., "Local search and metaheuristics for the quadratic assignment problem", Technical Report AIDA-01-01, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2001.
129. El-Ghazali, T. and Vincent, B., "COSEARCH: A parallel cooperative metaheuristic", Journal of Mathematical Modeling and Algorithms, Vol. 5, No. 1, pp. 5-22, April, 2006.
130. Lim, M., Yuan, Y. and Omatu, S., "Extensive Testing of a Hybrid Genetic Algorithm for Solving Quadratic Assignment Problems", Journal of Computational Optimization and Applications, Vol. 23, No. 1, pp. 47-64, October, 2002.
131. Mann, H. and Whitney, D., "On a test of whether one of 2 random variables is stochastically larger than the other", Annals of Mathematical Statistics, Vol. 18, No. 1, pp. 50-60, 1947.
132. Wilcoxon, F., "Individual comparisons by ranking methods", Biometrics Bulletin, Vol. 1, No. 6, pp. 80-83, 1945.
133. Shier, R., "The Mann-Whitney U Test", Mathematics Learning Support Centre, Loughborough Univ., England, Available at online at <http://mlsc.lboro.ac.uk/-resources/statistics/Mannwhitney.pdf>.
134. Liu, H., Abraham, A., and Zhang, J., "A Particle Swarm Approach to Quadratic Assignment Problems", 11th Online World Conference on Soft Computing in Industrial Applications (WSC11), Springer Verlag, Germany, September, 2006.
135. Pohlheim, H., "Documentation for GEATbx version 3.7: Genetic and Evolutionary Algorithm Toolbox for use with Matlab", Available at <http://www.geatbx.com>.
136. Hedar, A., *Studies on Meta-heuristics for Continuous Global Optimization Problems*, A PhD. Thesis, Kyoto University, Kyoto, Japan, June, 2004.
137. Test Functions for Unconstrained Global Optimization, available at [http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar\\_files/go.htm](http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm).
138. Liang, J., Qin, A., Suganthan, P. and Baskar, S., "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", IEEE Transactions on Evolutionary Computation Vol. 10, No. 3, pp. 281-295, 2006.



139. Bersini, H., Dorigo, M., Langerman, S., Seront, G. and Gambardella, L., "Results of the first international contest on evolutionary optimization (1st ICEO)", Proceedings of IEEE International Conference on Evolutionary Computation, pp. 611-615, May, 1996.
140. Mishra, S. K., "Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program", *Social Science Research Network* (SSRN), Working Papers Series, August, 2006, Available at <http://ssrn.com/abstract=924339>.
141. Parsopoulos, K.E. and Vrahatis, M.N., "Unified particle swarm optimization in dynamic environments", Lecture Notes in Computer Science, Springer Berlin / Heidelberg publisher, Vol. 3449, pp. 590 – 599, 2005.
142. Peram, T., Veeramachaneni, K., and Mohan, C., "Fitness-distance-ratio based particle swarm optimization", Proceedings of the IEEE Swarm Intelligence Symposium SIS '03, pp. 174-181, April, 2003.
143. Mendes, R., Kennedy, J. and Neves, J., "The fully informed particle swarm: simpler, maybe better", IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp. 204-210, June, 2004.
144. Van den Bergh, F. and Engelbrecht, A., "A Cooperative approach to particle swarm optimization", IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp. 225-239, June, 2004.
145. Vesterstrøm, J. and Thomsen, R., "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems", Proceedings of the 2004 Congress on Evolutionary Computation, Vol. 2, pp. 1980-1987, 2004.
146. Ratnaweera, A., Halgamuge, S. and Watson, H., "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients", IEEE Transactions on Evolutionary Computation Vol. 8, No. 3, pp. 240-255, 2004.
147. Hirsch, M., Meneses, C., Pardalos, P., and Resende, M., "Global optimization by continuous GRASP," 19th International Symposium on Mathematical Programming, Rio de Janeiro, Brazil, July 31 – August 4, 2006.

148. Song, Y., Chen, Z., and Yuan, Z., "New Chaotic PSO-Based Neural Network Predictive Control for Nonlinear Process", IEEE Transactions on Neural Networks, Vol. 18, No. 2, March, 2007.
149. Zhang, W.J., Xie, X.F., Yang, Z.L., "Hybrid particle swarm optimizer with mass extinction", Proceedings of the International Conference on Communication, Circuits and Systems, pp. 1170-1173, 2002.
150. Carlisle, A., and Dozier, G., "An Off-The-Shelf PSO", Proceedings of the 2001 Workshop on Particle Swarm Optimization, pp. 1-6, Indianapolis, IN, 2001.
151. Molga, M., and Smutnicki, C., "Test functions for optimization needs", Available at <http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf>.
152. Vesterstroem, S. and Riget, J., "Particle swarms: extensions for improved local, multi-modal, and dynamic search in numerical optimization." Master's thesis Department of Computer Science, University of Aarhus, 2002.
153. Parsopoulos, K. E. and Vrahatis, M. N., "Initializing the particle swarm optimizer using the nonlinear simplex Method," in Grmela, A. and Mastorakis, N. E. (eds.) Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation WSEAS Press, pp. 216-221, 2002.
154. Yao, X. and Liu, Y. "Fast evolutionary programming", Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96), pp.451-460, San Diego, CA, USA, 1996.
155. Hongbo, L. and Abraham, A., "Fuzzy Adaptive Turbulent Particle Swarm Optimization", International Journal of Innovative Computing and Applications, Vol. 1, No. 1, pp. 39-47, 2007.

## APPENDIX

### Matlab code for the proposed MCDM-PSO (QAP)

```
clear
rand('state',0) tt=cputime
itr=50% itr=10
P=30
nP=5
ci=30
for i=1:P;
    xpl(i,1:ci)=randperm(ci);
end
xpl;
x=[xpl];
for p=1:P
    for uuu=1:ci
        for vvv=1:ci
            td0(p)=dis(uuu,vvv)*yy(find(x(p,')==uuu),find(x(p,')==vvv));
            td1(uuu,vvv)=td0(p);
        end
    end
    td(p)=sum(sum(td1));
end
%
for p=1:P
    pbest(p)=abs(td(p));
    pbestx(p,1:ci)=x(p,1:ci);
end
s1=abs(td);
s=[s1' x(:, :)];
[u,v]=min(s(:,1));
gbest=u;
gbestx=s(v,2:ci+1);
%
for ii=1:itr
    A=[zeros(P,1) x];
    B=[zeros(P,1) x];
    for p=1:P
        for pp=1:P
            ld=leven_dis(A(p,:),B(pp,:));
            ldd(pp)=ld;
        end
        lddd(p,:)=ldd;
    end
    lddd;
    %building membership functions for fitness and edit distances

    limf1=gbest;
    limf2=max(td);
    limd1=min(nonzeros(lddd));
    limd2=max(nonzeros(lddd));
    %fitness membership function(decreasing is good (minimization))
```

```

xfitnes=limf2:(limf1-limf2)/99:limf1;
yfitnes=0:1/99:1;
mffitnes=fit(xfitnes',yfitnes','linearinterp');
% plot(xfitnes,yfitnes);%ploting membership function
feval(mffitnes,td);%evaluating membership for new fitness values
%
%edit distances membership function(increasing is good (promoting
diversity)
xdistance=limd1:(limd2-limd1)/99:limd2;
ydistance=0:1/99:1;
mfdistance=fit(xdistance',ydistance','linearinterp');
% plot(xdistance,ydistance);%ploting membership function
kki=feval(mfdistance,lddd);%evaluating membership for distance values
ghi=find(kki<0);
kki(ghi)=0;
rkki=reshape(kki,P,P);
%Neighborhoods best fitness and positions(neighborhoods are divided
every 5
%numbers)
nmmn=0;
for nesz=0:P/nP:P-(P/nP)
[n_best,n_best_ord]=min(pbest((1+nesz):(P/nP)+nesz));
ne_best_ord=n_best_ord+nesz;
nmmn=nmmn+1;
nei_best(nmmn)=n_best;%vector of the fitness(best) in every
neighborhood
nei_best_ord(nmmn)=ne_best_ord;%vector of orders of every best in every
neighborhood
for p=1+nesz:(P/nP)+nesz
A=[zeros(1,1) x(p,:)];
B=[zeros(1,1) pbestx(ne_best_ord,:)];
l1d=leven_dis(A,B);%Lev. edit distance between every individual and the
best of hi neighborhood
l1dd(p)=l1d;%all Lev. edit distance between every individual and the
best of his neighborhood
end
end

%calculations of lev. edit distance between every individual and its
best own experince positions
C=[zeros(P,1) x];
D=[zeros(P,1) pbestx];
fuz_r_total=cell(P,1);
for p=1:P
ld1=leven_dis(C(p,:),D((p),:));%lev. edit distance between every member
his own best position
lddd1(p)=ld1;
end

lddd1;

n_h_f_f(1:(P/nP))=nei_best(1);
n_h_f_f((P/nP)+1:2*P/nP)=nei_best(2);
n_h_f_f((2*P/nP)+1:3*P/nP)=nei_best(3);
n_h_f_f((3*P/nP)+1:4*P/nP)=nei_best(4);

```

```

n_h_f_f((4*P/nP)+1:5*P/nP)=nei_best(5);
cpo=zeros(P,ci);

for jil=1:P
    if jil<=(P/nP)
        cpo(jil,:)=pbestx(nei_best_ord(1),:);
    end
    if (P/nP)<jil&jil<=(2*P/nP)
        cpo(jil,:)=pbestx(nei_best_ord(2),:);
    end
    if (2*P/nP)<jil&jil<=(3*P/nP)
        cpo(jil,:)=pbestx(nei_best_ord(3),:);
    end
    if (3*P/nP)<jil&jil<=(4*P/nP)
        cpo(jil,:)=pbestx(nei_best_ord(4),:);
    end
    if (4*P/nP)<jil&jil<=(5*P/nP)
        cpo(jil,:)=pbestx(nei_best_ord(5),:);
    end
end

%creating random positions for play behavior
for i=1:P;
    xpr1(i,1:ci)=randperm(ci);
end
xpr1;
xr=[xpr1];
%fitness function calculations (distances)
for p=1:P
    for uuur=1:ci
        for vvvr=1:ci
            tdr0(p)=dis(uuur,vvvr)*yy(find(xr(p,')==uuur),find(xr(p,')==vvvr));
            tdr1(uuur,vvvr)=tdr0(p);
        end
    end
    tdr(p)=sum(sum(tdr1));
end

%edit distance between random position xr(play) and self position x
E=[zeros(P,1) x];
F=[zeros(P,1) xr];
for p=1:P
    ldr=leven_dis(E(p,:),F(p,:));
    lddr(p)=ldr;
end

lddr;
%yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

for p=1:P
    fuz_m1_retre=feval(mfdistance,ldddl(p));
    if fuz_m1_retre<0

```

```

        fuz_m1_retre=0;
    end
    fuz_m2_retre=feval(mffitness, pbest(p));

    if fuz_m2_retre<0
        fuz_m2_retre=0;
    end
    fuz_m1_imi=feval(mfdistance,l1dd(p));
    if fuz_m1_imi<0
        fuz_m1_imi=0;
    end
    fuz_m2_imi=feval(mffitness,n_h_f_f(p));

    if fuz_m2_imi<0
        fuz_m2_imi=0;
    end
%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
%play behavior
fuz_m1_play=feval(mfdistance,l1dd(p));
    if fuz_m1_play<0
        fuz_m1_play=0;
    end
    fuz_m2_play=feval(mffitness, tdr(p));

    if fuz_m2_play<0
        fuz_m2_play=0;
    end
%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
%momentum behavior (edit distance x,x is zero)- distance membership
grades are zeros
fuz_m2_mom=feval(mffitness, td(p));

    if fuz_m2_mom<0
        fuz_m2_mom=0;
    end
%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
    fuz_ranked=[sort([fuz_m1_retre fuz_m2_retre]);sort([fuz_m1_imi
fuz_m2_imi]);sort([fuz_m1_play fuz_m2_play]);sort([0 fuz_m2_mom])];
    fuz_r_total(p)=fuz_ranked;
    order_wights=[.8 .2];

    retre=[sum(fuz_ranked(1,:).*order_wights(1,:))];
    imi=[sum(fuz_ranked(2,:).*order_wights(1,:))];
    play=[sum(fuz_ranked(3,:).*order_wights(1,:))];
    mom=[sum(fuz_ranked(4,:).*order_wights(1,:))];

    odi_retre=floor((retre/(imi+retre+play+mom))*ci);
    odi_imi=floor((imi/(imi+retre+play+mom))*ci);
    odi_play=floor((play/(imi+retre+play+mom))*ci);
    odi_mom=floor((mom/(imi+retre+play+mom))*ci);

%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
%behaviors order
beh_ord=sort([retre imi play mom]);
imi_ord=find(beh_ord==imi);

```

```

beh_ord(imi_ord(1))=nan;
retre_ord=find(beh_ord==retre);
beh_ord(retre_ord(1))=nan;
play_ord=find(beh_ord==play);
beh_ord(play_ord(1))=nan;
mom_ord=find(beh_ord==mom);

%[mom_ord(1) play_ord(1) imi_ord(1) retre_ord(1)];
recu_mat=zeros(5,ci);

recu_mat(1,:)=x(p,:);
recu_mat(play_ord(1)+1,:)=xr(p,:);
recu_mat(imi_ord(1)+1,:)=cpo(p,:);
recu_mat(mom_ord(1)+1,:)=x(p,:);
recu_mat(retre_ord(1)+1,:)=pbestx(p,:);

rate_mat=zeros(4,1);
rate_mat(play_ord(1))=odi_play;
rate_mat(imi_ord(1))=odi_imi;
rate_mat(mom_ord(1))=odi_mom;
rate_mat(retre_ord(1))=odi_retre;

a=recu_mat(2,:);
b=recu_mat(1,:);
rate=rate_mat(1);
if rate==0
    rate=1;
end
imit
xx(p,1:ci)=b;

b=xx(p,1:ci);
a=recu_mat(3,:);
rate=rate_mat(2);
if rate==0
    rate=1;
end
imit
xxx(p,1:ci)=b;

b=xxx(p,1:ci);
a=recu_mat(4,:);
rate=rate_mat(3);
if rate==0
    rate=1;
end
imit
xxxx(p,1:ci)=b;

b=xxxx(p,1:ci);
a=recu_mat(5,:);
rate=rate_mat(4);
if rate==0
    rate=1;
end

```

```

    imit
    xxxxx(p,1:ci)=b;
end
x=xxxxx;
xold=x;
for p=1:P

    hj1=randint(1,1,[1 ci]);
    hj2=randint(1,1,[1 ci]);
    x(p,hj1)=xold(p,hj2);
    x(p,hj2)=xold(p,hj1);

end

%fine tuning
%=====
for ii=1:itr

    for p=1:P

        a=gbestx;

        b=x(p,1:ci);

        rate=28;
        if ii>=500
            rate=13;
        end
        imit
        xx(p,1:ci)=b;

    end
    for p=1:P
        b=xx(p,1:ci);
        a=pbestx(p,1:ci);

        rate=25;

        imit
        xxx(p,1:ci)=b;
    end
    for p=1:P
        b=xxx(p,1:ci);
        a=xx(p,1:ci);

        rate=5;
        imit
        xxxx(p,1:ci)=b;
    end

    x=xxxxx;
    xold=x;

    for p=1:P

```



```

    hj1=randint(1,1,[1 ci]);
    hj2=randint(1,1,[1 ci]);

    x(p,hj1)=xold(p,hj2);
    x(p,hj2)=xold(p,hj1);

    end
x;
for p=1:P
    for uuu=1:ci
        for vvv=1:ci

td0(p)=dis(uuu,vvv)*yy(find(x(p,:)==uuu),find(x(p,:)==vvv));
            td1(uuu,vvv)=td0(p);
        end
    end
    td(p)=sum(sum(td1));
end

for p=1:P
h(p)=abs(td(p));
if h(p)<= pbest(p);
    pbest(p)=abs(td(p));
    pbestx(p,1:ci)=x(p,1:ci);

end

end

s=[pbest' pbestx];

[u,v]=min(s(:,1));
gbest=u;

gbestx=s(v,2:ci+1);
gbestxx(1:ci,ii)=gbestx';
gbestt(ii)=[gbest];

[gbest,ord]=min(gbestt);
gbest
gbestx=gbestxx(1:ci,ord);
end
end
%=====
for p=1:P
    for uuu=1:ci
        for vvv=1:ci
            td0(p)=dis(uuu,vvv)*yy(find(x(p,:)==uuu),find(x(p,:)==vvv));
            td1(uuu,vvv)=td0(p);
        end
    end
end

```

```

        td(p)=sum(sum(td1));
    end

    for p=1:P
        h(p)=abs(td(p));
        if h(p)<= pbest(p);
            pbest(p)=abs(td(p));
            pbestx(p,1:ci)=x(p,1:ci);

        else
            pbest(p)=abs(td(p));
        end

    end

    end

    s=[pbest' pbestx];

    [u,v]=min(s(:,1));
    gbest=u;

    gbestx=s(v,2:ci+1);
    gbestxx(1:ci,ii)=gbestx';
    gbestt(ii)=[gbest];

    [gbest,ord]=min(gbestt);
    gbest

    gbestx=gbestxx(1:ci,ord);

    % x=s(:,2:(ci+1));%in case of q>1

    ttt=cputime-tt

```

### Levenshtein edit distance Function

```
function [L_d]=leven_dis(x,y);

n=length(y);
m=length(x);

z=zeros(m,n);
for n=2:n;
    z(1,n)=n-1;
end
for m=2:m;
    z(m,1)=m-1;
end

for j=2:m
    for i=2:n
        x(j);
        y(i);
        if x(j)==y(i)
            cost=0;
        elseif x(j)~=y(i)
            cost=1;
        end

        z1=min(z(j,i-1)+1,z(j-1,i)+1);
        z(j,i)=min(z1,z(j-1,i-1)+cost);
    end

end

x;
y;

z;
distance=z(i,j);

L_d=distance;
x;
y;
```

### Items transmission routine

```
rnd_imit=randint(1,rate,[1,(ci-1)]);
t=unique(rnd_imit);% not to repeat paths
[y,Z]=size(t); %Z=no.of colums

bo=b;
for z=1:Z
    b(t(z))=a(t(z));
    v=find(bo==b(t(z)));
    b(v)=bo(t(z));
    bo=b;
    bo=b;
end
```

### Matlab code for continuous optimization problems

```
clear
format long;
warning('off');
for num_run=1:1
t=cputime;
itr=2000
P=30
c2=0;
c1=0;
Q=1
pou=1%weight of gbestx in randx (normal average when pou=1)
muo=10;%std the random position from the mean (pbest,ne_best , global
best)
mom_rate=100;
ci=5; %equals d the deimension
nP=5;
tol=0;
prob_no=4%for ICEO functions
xmin=0
xmax=pi
x=unifrnd(xmin,xmax,[ci P]);
xp=x;
%=====
for p=1:P
f(p)=iceo(prob_no,x(:,p));
end

for p=1:P
pbest(p)=(f(p));
for d=1:ci
pbestx(d,p)=x(d,p);
end
end

s=[(f);x(:, :)];

[u,v]=min(s(1,:));
gbest=u;
gbestx=s(2:ci+1,v);

pbest=s(1,:);
pbestx=s(2:ci+1,:);
int_gbest=u;
for ii=1:itr

%neighborhood best experience preparation(imitation)[matrix cpo has
all information positions and previous best solutions for neighbors]
nmnm=0;
for nesz=0:P/nP:P-(P/nP)

[n_best,n_best_ord]=min(pbest((1+nesz):((P/nP)+nesz)));
```

```

ne_best_ord=n_best_ord+nesz;
nmmn=nmmn+1;
nei_best(nmmn)=n_best;%vector of the previous fitness(best) in every
neighborhood
nei_best_ord(nmmn)=ne_best_ord;%vector of orders of every best in every
neighborhood
pbestx_ne=[pbestx(:,nei_best_ord)];
end
n_h_f_f(1:(P/nP))=nei_best(1);
n_h_f_f((P/nP)+1:2*P/nP)=nei_best(2);
n_h_f_f((2*P/nP)+1:3*P/nP)=nei_best(3);
n_h_f_f((3*P/nP)+1:4*P/nP)=nei_best(4);
n_h_f_f((4*P/nP)+1:5*P/nP)=nei_best(5);

%construct full matrix of neighbors previous best and corresponding
%position
cpo=zeros(ci+1,P);

for jil=1:P
    if jil<=(P/nP)
        cpo(1,jil)=nei_best(1);
        cpo(2:ci+1,jil)=pbestx_ne(:,1);
    end
    if (P/nP)<jil&jil<=(2*P/nP)
        cpo(1,jil)=nei_best(2);

        cpo(2:ci+1,jil)=pbestx_ne(:,2);
    end
    if (2*P/nP)<jil&jil<=(3*P/nP)
        cpo(1,jil)=nei_best(3);

        cpo(2:ci+1,jil)=pbestx_ne(:,3);
    end
    if (3*P/nP)<jil&jil<=(4*P/nP)
        cpo(1,jil)=nei_best(4);

        cpo(2:ci+1,jil)=pbestx_ne(:,4);
    end
    if (4*P/nP)<jil&jil<=(5*P/nP)
        cpo(1,jil)=nei_best(5);

        cpo(2:ci+1,jil)=pbestx_ne(:,5);
    end
end

end
%=====

%=====

for ioo=1:P
    for joo=1:ci

```

```

randx(joo,ioo)=normrnd((pbestx(joo,ioo)+pou*s(joo+1,v)+cpo(joo+1,p))/(p
ou+2),muo);%-15*rand+15*rand;
    while randx(joo,ioo)<xmin|randx(joo,ioo)>xmax

randx(joo,ioo)=normrnd((pbestx(joo,ioo)+pou*s(joo+1,v)+cpo(joo+1,p))/(p
ou+2),muo);%-15*rand+15*rand;
    end
end
end

%=====

for p=1:P
f_play(p)=iceo(prob_no,randx(:,p));
end
%=====

%fuzzy memberships
d_play=abs(randx-x);
d_imit=abs(cpo(2:ci+1,:)-x);
d_mem=abs(pbestx-x);
dmax1=max(max(d_play));
dmax2=max(max(d_imit));
dmax3=max(max(d_mem));
dmax_pre=max(dmax1,dmax2);

d_maxlimit=max(dmax_pre,dmax3);%max distance amongst all

fmin1=min(f_play);
fmin2=min(cpo(1,:));
fmin3=min(pbest);
fmin_pre=min(fmin1,fmin2);
fmax1=max(f_play);
fmax2=max(cpo(1,:));
fmax3=max(pbest);
fmax_pre=max(fmax1,fmax2);
f_maxlimit=max(fmax_pre,fmax3);

for p=1:P
for d=1:ci
    alfa1(d,p)=abs((cpo(d+1,p)-x(d,p))/d_maxlimit;
    beta1(d,p)=-(cpo(1,p)/(f_maxlimit-0))+1;
    sort1=sort([alfa1(d,p) beta1(d,p)]);
    rindx1(d,p)=sum(sort1.*[0.8 0.2]);
end
end

```

```

for p=1:P
for d=1:ci
    alfa2(d,p)=abs((pbestx(d,p)-x(d,p)))/d_maxlimit;
    beta2(d,p)=-(pbest(1,p)/(f_maxlimit-0))+1;
    sort2=sort([alfa2(d,p) beta2(d,p)]);
    rindx2(d,p)=sum(sort2.*[0.8 0.2]);
end
end

for p=1:P
for d=1:ci
    alfa3(d,p)=abs((randx(d,p)-x(d,p)))/d_maxlimit;
    beta3(d,p)=-(f_play(1,p)/(f_maxlimit-0))+1;
    sort3=sort([alfa3(d,p) beta3(d,p)]);
    rindx3(d,p)=sum(sort3.*[0.8 0.2]);
end
end

for p=1:P
for d=1:ci
    rindx_imi(d,p)=rindx1(d,p)/(rindx1(d,p)+rindx2(d,p)+rindx3(d,p));
    rindx_mem(d,p)=rindx2(d,p)/(rindx1(d,p)+rindx2(d,p)+rindx3(d,p));
    rindx_play(d,p)=(rindx3(d,p)/(rindx1(d,p)+rindx2(d,p)+rindx3(d,p)));
end
end

for p=1:P
    rindx_mom=ones(ci,P);%has no influence in the algorithm

w=(rand^mom_rate)*(xmax-xmin); %momentum influence
w1=(rand^ii)*(xmax-xmin);
%w1=.9-(ii*(0.9-.4)/itr);
for d=1:ci
x(d,p)=(rindx_mom(d,p)*x(d,p))+(w)+(rindx_imi(d,p)*(cpo(d+1,p)-
x(d,p)))+(rindx_mem(d,p)*(pbestx(d,p)-
x(d,p)))+(rindx_play(d,p)*(randx(d,p)-x(d,p)))+(c2*rand(1)*(gbestx(d)-
x(d,p)));
%x(d,p)=(x(d,p))+(w1)+(c1*rand(1)*(pbestx(d,p)-
x(d,p)))+(c2*rand(1)*(gbestx(d)-x(d,p)));

if x(d,p)<xmin|x(d,p)>xmax
    x(d,p)=pbestx(d,p); %xp
end
end

end
xp=x;

for p=1:P
f(p)=iceo(prob_no,x(:,p));
end

%=====

```



```

for p=1:P
h(p)=(f(p));
if h(p)<= pbest(p);
    pbest(p)=(f(p));
    pbestx(:,p)=x(:,p);

end

end

s=[pbest;pbestx];
pbest=s(1,:);
pbestx=s(2:ci+1,:);
[u,v]=min(s(1,:));
gbest=u

gbestx=s(2:ci+1,v);
gbestt(ii)=[gbest];

gbestx=s(2:ci+1,v);

end
gbesttt(num_run)=gbest
end

extime=cputime-t;

```