Multiagent Coordination Using Probability Collectives

Lutfia Khalifa Haj Mohamed

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2017
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Director

I certify that this thesis satisfies the requirements as thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Adnan Acan
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Mehmet Bodur
2. Asst. Prof. Dr. Adnan Acan
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

This thesis motivates and describes the use of probability collectives (PC) with a multiagent coordination system to solve different problems. The main challenge was to enable the agents to work in a coordinated way, optimizing the local utilities and contributing the maximum or minimum towards optimisation of a global objective. The approach was validated solving numerical benchmark problems such as sphere function in which the coupled variables are seen as autonomous agents working collectively to achieve the optimum solution. Moreover, PC algorithm solved successfully repeated games such as prisoner's dilemma, stag hunt, the battle of sexes game and choose sides. In all experimental trials, the optimum results were obtained at a reasonable computational cost.

Keywords: Probability Collectives, Collective intelligence, Multiagent systems, Game theory.

ÖZ

Bu tez farklı problemleri çözmek için olasılık derlemelerinin (PC) çok ajanlı koordinasyon sistemi ile kullanımını motive eder ve açıklar. Ana zorluk, ajanlarin koordineli bir şekilde çalışmasını sağlamak, yerel memniyetin en iyilenmesini sağlamak ve küresel bir hedefin maksimize edilmesine katkıda bulunmaktır. Bu yaklaşım in başarimi değişkenlerin, en iyi çözümü elde etmek için birlikte çalışan özerk ajanlar olarak görülen küre işlevleri gibi sayısal karşılaştırma problemlerini çözerek gösterilmiştir. Buna ek olarak, PC algoritması esirlerin ikilemleri, haydut avı, cinsiyetler savaşı gibi tekrarli oyunlarda en iyi stratejilerin bulunmasi icin kullanildi. Tüm deneysel denemelerde, en iyi sonuçlar makul bir hesaplama maliyetiyle elde edilmiştir.

Anahtar Kelimeler: Olasılık Kolektifleri, Kollektif Zeka, Çok ajanli Sistemler, Oyun Teorisi.

DEDICATION

To my beloved mother

ACKNOWLEDGMENT

At first, I thank Allah for giving me the strength and potential to continue, guidance and opportunity to pursue the present study up to a concluding level. Secondly, I would like to express my special appreciation and thanks to my advisor Asst. Prof. Dr. Adnan Acan, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for promoting me to become a researcher. His priceless pieces of advice in my research have always been my guidelines.

I also thank Asst. Prof. Dr. Ahmet Ünveren for his support and motivation.

My special thanks go to my family. Words cannot express how grateful I am to my mother, my sisters for all of the sacrifices they have made on my behalf. Your prayer for me was what sustained me thus far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimisation
BFGS	Broyden Flecther Goldfarb Shannon
COIN	Collective Intelligence
DEA	Differential Evolution Algorithm
GA	Genetic Algorithm
MAS	Multiagent System
MCOP	Multiagent Constraint Optimisation Problem
NNDS	Nearest Newton Descent Scheme
PC	Probability Collective
PSO	Particle Swarm Optimisation
SA	Simulating Annealing
SI	Swarm intelligence

Chapter 1

INTRODUCTION

1.1 Motivation

There is a great interest in the field of collective intelligence (COIN) due to its wide applications in many areas such as computer network and collective robotics as well as applications on the internet, games and movies. COIN framework consists of a huge number of autonomous agents, interacting locally both among themselves as well as an active environment, which comes out of the collaboration and competition of many individuals. These individuals are self-interested in some specific direction to select their actions and receive remunerations depending on a utility function. Moreover, the process repeats and converges to equilibrium when there is no increase in rewards for the agents through trying a variable. This concept is called a Nash equilibrium (EN). Hence, the concept of probability collectives (PC) becomes the implementation of the concept of Nash Equilibrium successfully [1] [2].

Fundamentally, PC algorithm is a modern method to solve distributed optimisation problems. The PC approach has strong connections to Game Theory, Statistical Physics and Optimisation [2][13]. In PC, the variables are denoted as individual agents/players and the distributed optimisation problem is considered as a game played by these agents [5] [3]. Theory of PC allocates probability distributions to select the agents' moves and allows each agent to autonomously update its own probability distribution at each iteration. These agents select the particular action based on the highest probability to optimise its own utility function. Thus, the algorithm continues to find the best solution until the convergence reaches to the globally optimal solution or one of stopping criteria such as $T = 0$ is achieved[4].

1.2 Advantages of Probability Collectives

Probability collectives algorithm has many benefits over the other optimisation techniques that can be used for the solution of numerical problems:

- In PC, every agent autonomously updates its own probability distribution parameters iteratively, and it can be used on continuous, discrete or mixed variables [4] [6].

- A set of probability strategies that is a vector of real numbers permits the technique of optimisation using Euclidean vectors [6].

- The cost function of PC can be irregular or noisy because PC is a robust algorithm [6].

- A variable with a peaky distribution plays a more significant role in the optimisation task than a variable with a broad distribution since PC provides the sensitivity information about the problem [3].

- Each agent (variable) can find the minimum value of the global objective function by using a Homotopy function that is easier to compute and optimize [7].

1.3 Thesis Work

Many types of methods are used to solve distributed optimisation problems by using variety techniques such as genetic algorithms (GA), particle swarm optimisation (PSO), and simulating annealing (SA). In this thesis, we will use probability collectives to solve unconstrained and constrained optimisation problems in order to make convergence more rapid as well as reducing the computational cost. We will apply PC to solve the numerical benchmark problems, El Farol bar problem, Multi-agent coordination as a case study to evaluate the N-queens problem, and investigating the evolution of cooperation in repeated games (Prisoner's dilemma, Stag hunt, Battle of sexes game, and Choose sides).

1.4 Outline of This Thesis

The remainder of this thesis is organised as follows:

Chapter 2

- Background information in the fields of optimisation and multiagent systems.
- Literature review of probability collectives.

Chapter 3

- Details of probability collectives algorithm.

Chapter 4

- Describes the specific implementation of probability collectives to solve Benchmark, N-queen, El Farol Bar problems, and Repeated games.

Chapter 5

- Displays the experimental evaluations of probability collectives algorithm, together with detailed discussions on the obtained results.

Chapter 6

- Presents the conclusion and future work.

Chapter 2

LITERATURE REVIEW

2.1 Introduction to Optimisation

In the simplest case, an optimisation problem consists of maximising or minimising a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalisation of optimisation theory and techniques to other formulations comprises a large area of applied mathematics. Hence, optimisation problems involve searching for a set of potential solutions satisfying a number of pre-specified criteria. One of the hardness in solving the real-world optimisation problems is that they have a variety of forms and kinds. Some problems have only one objective to optimise while some others may have multi-objectives. Additionally, some problems may be highly constrained and some have multiple optimal solutions [9].

2.1.1 Numerical Function Optimisation Problems

Numerical Function optimisation problems can be formulated in a standard generic form as follows:

Minimise/Maximise $F(x)$ (2.1)

Subject to

Equality constraints $h_i(x) = 0\,, \quad i = 1, \dots, n$

Inequality constraints $g_j(x) \le 0\,, \quad i = 1, 2, .., m$

$x_i^l \le x_i \le x_i^u$, $j = 1,2, \dots \dots \dots \dots \dots \dots \dots .. k$

where x is a vector of k decision variables:$(x_1, x_2, \ldots x_k)^T$. The decision variable space is limited by a set of boundary constraint where $[x_i^l, x_i^u]$ are lower and upper bounds for the decision variables. All solutions that satisfy all constraints and variable bounds are called feasible solutions. Otherwise they are called infeasible solutions.

2.1.2 Global and Local Optimal Solutions

A considering a minimisation problem, objective function F has a local minimum point at the point x^* if:

$$G(x^*) \leq G(X) \text{ For all feasible } X$$

This means a local optimum is a solution, which is optimal (either maximal or minimal) within a neighbouring set of candidate solutions.

The objective function F has a global minimum point at the point X^* if:

$$G(X^*) \leq G(X) \text{ For all X in the feasible region}$$

Hence, a global optimum is an optimal solution among all possible solutions, not just those in a particular neighbourhood of values. These concepts are shown in Figure 2.1.

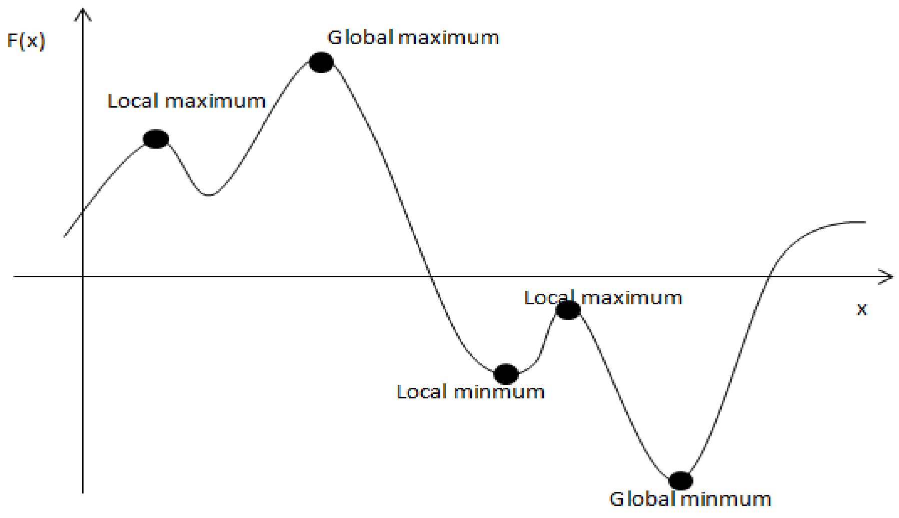


Figure 2.1: Minimum and Maximum Points

2.1.3 Nature-Inspired Optimisation Approaches

Many nature-/bio-inspired optimisation techniques have been evolved in the past few years such as Evolutionary Algorithm (EA), and Swarm Intelligence (SI) which have been developed. For example, the Genetic Algorithm (GA) works on the principle of Darwinian Theory of survival of the fittest in population. According to [26] and [27], the population is developed based on some operators like selection and crossover. GA may coverage very close to the global optimum. Similarly, Differential Evolution (DE) was proposed by Storn and Price, which helps to explore and locally exploit the decision space to reach the global solution. Although, easy to implement, there are many problem dependent parameters required to be tuned and may also require several associated trials to be performed.

Theories have been inspired by social behaviour of nature organisms such as fish, bees, and birds, which can interact with one another. The model of Swarm intelligence is a decentralised optimisation approach. In SI, each agent improves itself by sharing the information with others in an environment. SI such as Particle Swarm Optimisation (PSO) is inspired by social of birds flocking and schooling of fish seeking for food. The Ant Colony Optimisation (ACO) relies on the ants' social behaviour for foraging food by following the smallest track. As the same ACO, the Bee Algorithm tends to optimize the utilization of a number of members, which are included in specific predefined tasks. Generally, the swarm techniques are computationally intensive [8].

2.2 Introduction to Multi-agent Systems (MAS)

What are Multi-agent Systems?

Multi-agent systems consist of a number of autonomous agents, which interact with each other or with their environment to perform some set of tasks or to satisfy some set of goals. One of the main goals of MAS is to find solutions to complex systems problems and to deal with tasks that are beyond the ability of a single agent [10]. Figure 2.2 shows a generic description of a multi-agent system.

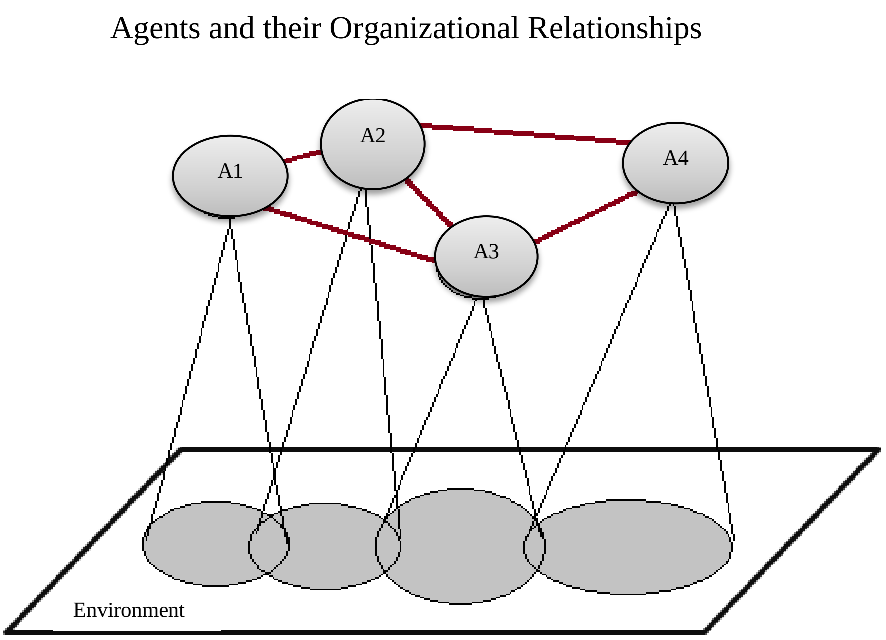


Figure 2.2: Generic Description of A Multi-Agent System [10]

2.2.1 Advantages of Multi-agent Systems

Multi-agent systems have some benefits that can be listed as follows:

1- Due to parallel computation and asynchronous operation, MAS increase the speed and efficiency of operation [11].

2- MAS reduce computational cost due to individual agents; cost is much less than that of a centralised architecture [11].

3- MAS increase the reliability and robustness of the systems [11].

2.2.2 Study of Multi-agent Systems

There are many topics to study in multi-agent systems such as Agent-oriented software engineering, coordination, cooperation, and organisation. In this thesis, we will discuss the example of multi-agent coordination.

- **Multi-agent Coordination**

Multiple agents need many practical settings in order to coordinate their actions. This coordination includes combined decisions about resource distribution, scheduling, and planning that can be formulated as constraint optimisation problems [12]. We thus extend the standard definition of constraint optimisation to the multi-agent setting as follows:

Definition 1: constraint optimisation problem (COP) is a tuple (X,D,C,A) where:

$X = \{x_1, \ldots., x_M\}$ is a set of variables/agents.

$D = \{d_1, \ldots., d_m\}$ is a set of domains of the variables.

$C = \{c_1, \ldots., c_m\}$ is a set of constraints.

$R = \{r_1, \ldots., r_m\}$ is a set of relations, where a relation r_i is a function is assigned to each combination of values of the involved variables/agents.

All variables have values that satisfy all constraint and maximise the sum of agent utilities in MCOP as expresses by their relations. Note that variable, domains and constraints are common and agreed upon knowledge among the agents. On the other hand, the individual agents specify relations, and they do not necessarily have to report them correctly [12].

2.3 Literature Review of Probability Collectives

In few past years, there are challenges to solve complex problems by using a set of distributed intelligence agents. These agents act in some specific direction to find local payoffs or the best solution. Hence, Probability collectives (PC) in the framework of COIN are a new distributed optimisation algorithm that was first introduced by Dr David Worlpert in 1999 in a Technical Report suggested to NASA [7] [13]. In 2004, many modifications and applications have already been evolved, where Lee and Worlpert change the word utility with the private utility to reduce the sample size utilised in the PC algorithm with no prejudice and low contrast. The sample size has effectively been reduced using the data aging technique (Beneniawski et al 2005). Kulkarni et al (2008) shorted the sample region around the present optimal points by modifying the sample principle on original monte-carlo. In 2011, Worlpert et al have been updated some of the strategies like Steepest Decent, Nearest Newton, and Brower Fixed Point method [14]. Moreover, Worlpert et al used the significance sampling and parametric machine learning technique to classify the PC algorithm as 'Delayed Sampling' and place forward another 'Immediate Sampling'. Kulkarni et al. (2011) have used the Broyden Fletcher Goldfarb Shanno method (BFGS) to optimise objective system [14][13].

There are some applications of the PC algorithm. Numerical results on a set of benchmark functions demonstrate that PC method outperforms the GA algorithm in the rate of descent (Hunag et al. 2005). PC has been used in some combinatorial problems such as multiple travelling salesman problem (Kulkarni et.al 2010), school table scheduling problem (Autry and Brian 2008) and vehicle routeing problems (Kulkarni et.al 2010).

2.3.1 Probability Collectives Framework

The Probability collectives deal the variables in optimisation problem as agents/players. These agents make a decision to optimise the local rewards and the system level performance. Furthermore, agents determine their strategies of probability values. During its iterations, the agent updates its own probability to select specific actions on the base of the highest probability, in order to improve its own private utility. Eventually, the process continues, until the convergence reaches the global optimal or some stopping criteria such as temperature rate ($T = 0$) or no change in the objective function. More details of PC are given in Chapter 3.

2.3.2 Problems Definition

In this thesis, we will solve Numerical optimisation benchmarks, N-queens, El Farol bar, and repeated games problems using probability collectives algorithm.

A- Numerical Optimisation Benchmarks Problems

Twenty-three benchmark functions are classified into three categories based on their characteristics, which we will use in this thesis.

- Unimodal Functions

This category consists of functions ($F_1 - F_7$). Each function has only one global minimum and is high dimensional [15].

- High –Dimensional Multimodal Functions

This group involves functions ($F_8 - F_{13}$). There are several local minimums in each of the functions and they are high dimensional. This category is the most difficult problems compared with the previous group [15].

- Low-Dimensional Multimodal Functions

This category includes functions ($F_{14} - F_{23}$). They contain fewer local minimums and they are low Dimensional compared with the second group [15]. Table (2.1) shows the descriptions of each function which starts from Sphere function and goes down Shekel-10 function.

Table 2.1: The 23 Numerical Optimisation Benchmarks

Name	Test Function	N	Domain	Optimum
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0
Schewefel 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]^n$	0
Schewefel 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100,100]^n$	0
Schewefel 2.21	$f_4(x) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	$[-100,100]^n$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	30	$[-30,30]^n$	0
Step	$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100,100]^n$	0
Quartic	$f_7(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1)$	30	$[-30,30]^n$	0
Schwefel 2.26	$f_8(x) = -\sum_{i=1}^{n} (x_i \sin(\sqrt{x_i})$	30	$[-500,500]^n$	-12569.5
Rastrigin	$f_9(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12,5.12]^n$	0
Ackley	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \mathrm{EXP}\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i\right) + 20 + e$	30	$[-32, 32]^n$	0
Griewank	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^n$	0
Levy	$f_{12}(x) = \frac{\pi}{n}\left\{10sin^2(\pi y_1) + \sum_{i=1}^{29} (y_i - 1)^2[1 + 10sin^2(\pi y_{i+1}] + (y_n - 1)^2\right\} + \sum_{i=1}^{30} u(x_i, 10,100,4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k,) = \begin{cases} k(x_i - a)^m & ,x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m & ,x_i < -a \end{cases}$	30	$[-50,50]^n$	0

Levy 8	$f_{13}(x) = 0.1\Big\{sin^2(\pi 3x_1) + \sum_{i=1}^{29}(x_i-1)^2\,[1 + sin^2(3\pi x_{i+1})] + (x_n-1)^2[1 + sin^2(3\pi x_{30})]\Big\} + \sum_{i=1}^{30} u(x_i,5,100,4)$	30	$[-50,50]^n$	0
Shekel Foxholes	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	$[-65.536,65.536$	1
Kowalik	$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]^n$	0.0003075
Six-Hump Camel	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]^n$	-1.0316285
Branin	$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5,10] \times [0,15]$	0.398
Goldstein	$f_{18}(x) = [1 + (x_1+x_2+1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1-3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2,2]^n$	3
Hartman 4	$f_{19}(x) = -\sum_{i=1}^{4} c_i\, exp\left[-\sum_{j=1}^{4} a_{ij}\left(x_j-p_{ij}\right)^2\right]$	4	$[0,1]^n$	-3.86
Hartman 6	$f_{20}(x) = -\sum_{i=1}^{4} c_i\, exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j-p_{ij}\right)^2\right]$	6	$[0,1]^n$	-3.32
Shekel 5	$f_{21}(x) = -\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10
Shekel 6	$f_{22}(x) = -\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10
Shekel 10	$f_{23}(x) = -\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	-10

B- El Farol Bar Problem (EFBP)

El Farol Bar Problem was first proposed by W. Brian Arthur in 1994 based on a bar in Santa Fe New Mexico.

Figure 2.3: El Farol Bar in Santa Fe New Mexico[16]

The idea of this problem is that, there are N agents/people. Each Thursday, the agent tries to predict the bar's attendee "go to a bar or stay at home". If attendances are more than 60 of agents/people, the bar is crowded and then agent decides to stay at home otherwise attends the bar. Each agent/person cannot communicate with others, so no one has any information about the intentions of the others. They only have access to the number of the clients of the last weeks [16]. This problem can be formalised as follows.

- N is the number of agents/people.
- $H = [h_1, \dots . h_s]$ history list stores previous attendance at the bar, where history list size =$m \times 2$, where m is memory size.

- $s = [c, w_{1,} \ldots\ldots\ldots\ldots w_l]$ is list strategies for each agent which consist of a list of real valued numbers, where $[\, w_{1,} \ldots\ldots\ldots\ldots w_l]$ are weights that take a random value between -1 to +1, C is a constant.

- Calculate prediction using this formula:

$p(t) = c + \sum_{i=1}^{l} w_i a(t-i)$ (2.1)

Where $p(t)$ is the prediction for the attendance at week t, and $a(t-i)$ is actually recorded attendance at the week $(t-i)$.

- Find the best strategy for each agent for the next prediction using strategy score, which is the sum of the difference between its predictions and the actual attendees, this is known as error function:

$f(s,t) = \sum_{i=t-1}^{t-1} |p(s,i) - a(i)|$ (2.2)

- Select the best strategy based on minimum score (error) and use this strategy for predicting next week.

C-N-Queens Problem

The N-queens problem has been proposed by Max Bezzel in 1848 for normal 8×8 chessboard, which belongs to the class of constraint satisfaction problems. The objective of the problem is to put N queens on a $N \times N$ chessboard where there are no conflicts among any of queens such as no shared rows, columns, diagonals [17]. This problem is formalized as follows.

- Let $V = \{v_1, \ldots \ldots .. v_n\}$ is a group of variables and each of them is identical to a row in the chessboard [17].

- N is the number of queens.

- Every variable v_i takes a value from the domain where $D_i = \{1,2,3 \dots \dots \dots \dots N\}$, where every v_i corresponds to a column of the chessboard which we can put a queen [18].
- The constraints are $v_i \neq v_j$, $v_i - v_j \neq i - j$, $v_i - v_j \neq j - i$ [18].
- The objective function is non-attacking queens on the $N \times N$ chessboard by considering the chess rules.

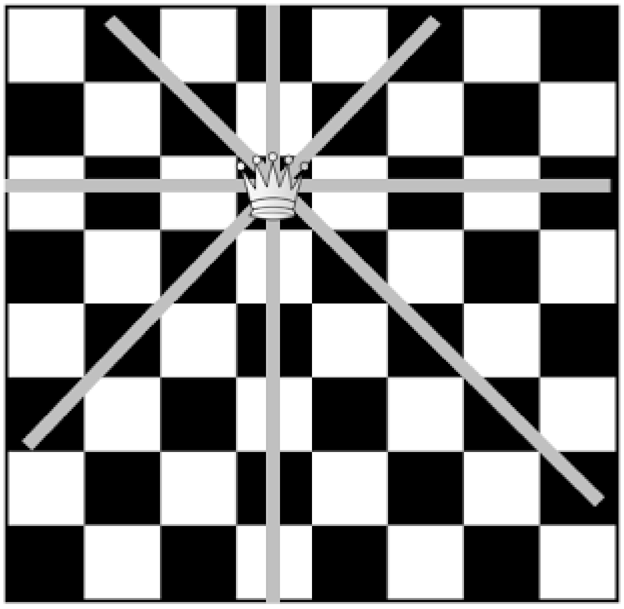


Figure 2.4: Example of Queen's Move [18]

D-Repeated Games

When the game is repeated a number of times by the same player, the game is called iterated game. Each player determines their strategies taking into account the effect of his previous actions on the future actions of others. In a repeated game, an equilibrium that is not stable may become stable, because the player will play the game again with the same player. Hence, Nash equilibrium is a set of strategies, each player is considered that no player has the incentive to change its strategy given what the other players are doing. The basic descriptions of games (prisoner’s dilemma, stag hunt, the battle of sexes game, choose sides) are as follows:

In the prisoner’s dilemma, two criminals are arrested and brought into a police station. The police think they committed a more serious crime but they do not have enough evidence to convict them. They need a confession. They take them and put them in separate rooms, so they can’t talk to each other. The police give each of them a choice. Admit your partner committed the crime and you will be free, but your partner will spend 5 years in a jail. If you do not confess and your partner does, it will be the reverse. On the other hand, if they both defect, they will spend only one year in jail. While if they both do not defect, they will serve three years. There is one pure strategy in this game when both player defect. This can be expressed in a normal form, which is shown in table 2.5 [20][21].

Table 2.2: Payoffs Matrix of the Prisoner’s Dilemma

		Player 2	
		Cooperate	**Defect**
Player 1	**Cooperate**	3,3	0,5
	Defect	5,0	1,1

The game of stag hunt involves two hunters. They can hunt a stag or a rabbit. If both hunters cooperate together, they hunt the stag. However, if hunters hunt separately, they will get the rabbit. This game has two pure strategies of Nash equilibrium, which differ from prisoner’s dilemma. So you can see in table 2.3 that if both players choose the stag then they will both get payoffs 2. This strategy is called payoffs dominant. On the other hand, if both players play alone they will both get payoffs 1. This strategy is known risk dominant [22].

Table 2.3: Payoffs Matrix of Stag Hunt

		Hunter 2	
		Stag	Rabbit
Hunter 1	Stag	2,2	0,1
	Rabbit	1,0	1,1

The battle of the sexes game includes a husband and his wife that want to spend a night out at either the opera or a football game. The husband would like to go to the football game, while the wife prefers to go to the opera. However, both prefer to go out together rather than alone. This game has two version. Version 2 is introduced to an account for the fact that a couple could choose the event that is not there the most preferred, while version 1 does not. Version 1 is called the battle of the sexes 1, whereas version 2 is called the battle of the sexes 2. This game is similar to the stag hunt, which has two pure strategies. These strategies occur when both go to the football game and both go to the opera [23]. The payoffs matrix is shown in table 2.4.

Table 2.4: Payoffs Matrix of The Battle of Sexes Game

		Wife	
		Opera	Football
Husband	Opera	3, 2	0, 0
	Football	0, 0	2, 3

a. The battle of the sexes 1

		Wife	
		Opera	Football
Husband	Opera	3, 2	1, 1
	Football	0, 0	2, 3

b. The battle of the sexes 2

Choose sides are a coordinate game, which involves two drivers. They drive along a dirt road. Both must swerve to avoid a head-on collision. If both choose the same sides (both left or both right), they will manage to pass each other. However, if they choose different sides such as (left, right), they will collide. This game has two pure strategies either both swerve to the left, or both swerve to the right. Both solutions are payoffs dominant. In table 2.5 the payoffs matrix of this game is shown clearly.

Table 2.5: Payoffs Matrix of The Choose sides

		Driver 1	
		Left	**Right**
Driver 2	**Left**	**1, 1**	**0,0**
	Right	**0,0**	**1,1**

You can see a summary of characteristics of these games in table 2.6.

Table 2.6: The Summary of Game Characteristics

Game	**Number of Pure Strategy Nash Equilibrium**	**Pure Strategy Nash Equilibrium**	**Zero Sum**
Prisoner's Dilemma	1	(Defect, Defect) = (1,1)	No
Stag Hunt	2	(Stag,Stag) = (2,2) (Rabbit, Rabbit) = (1,1)	No
Battle of the sexes 1 and 2	2	(Opera,Opera) = (3,2) (Football, Football)=(2,3)	No
Choose sides	2	(Left,Left) = (1,1) (Right,Right) = (1,1)	No

Chapter 3

Probability Collectives Algorithm

3.1 Probability Collectives Formulation

Probability collectives algorithm formalise as a group of N agents, each agent x_i can take on a finite number of values from interval $\Omega \in [x_i^L, x_i^H]$ and builds a set of solution through a strategy set x represented as [4][13]:

$$X_i = \left\{X_i^{[1]}, X_i^{[2]}, \dots \dots \dots, X_i^{[m_i]}\right\}, \quad i \in \{1,2, \dots \dots .., N\} \tag{3.1}$$

Where m_i is the number of strategies and N is the number of variables. Each agent i combines a set of strategies Y_i^r with cardinality m_i in cooperation with other agents as:

$$Y_i^{[1]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots .., X_i^{[1]}, \dots \dots .., X_{N-1}^{[?]}, X_N^{[?]}\right\} \tag{3.2}$$

The superscript [?] denotes to random selection and each agent has formed one strategy set for each remaining strategies of its strategy set X_i. Accordingly, the set of solutions build by agent i as shown below.

$$\begin{gathered}
Y_i^{[2]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots .., X_i^{[2]}, \dots \dots .., X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
Y_i^{[3]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots .., X_i^{[3]}, \dots \dots .., X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
\Downarrow \\
Y_i^{[r]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots .., X_i^{[r]}, \dots \dots .., X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
\Downarrow \\
Y_i^{[m_i]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots .., X_i^{[m_i]}, \dots .., X_{N-1}^{[?]}, X_N^{[?]}\right\}
\end{gathered} \tag{3.3}$$

As the same, all the remaining agents form their collective strategy sets as shown in equation (3.3), then every agent i evaluates the objective function for each of their combined strategy set $Y_i^{[m_i]}$ as:

$$\left[G\left(Y_i^{[1]}\right), G\left(Y_i^{[2]}\right), \dots\dots, G\left(Y_i^{[r]}\right), \dots\dots\dots, G\left(Y_i^{[m_i]}\right)\right] \tag{3.4}$$

Each agent finds the sum of the objective function for its combined strategy set to be minimised as follows [13]:

$$\left.\begin{array}{l}
Y_1^{[1]} = \left\{X_1^{[1]}, X_2^{[?]}, \dots\dots, X_i^{[?]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_1^{[1]}\right) \\
Y_1^{[2]} = \left\{X_1^{[2]}, X_2^{[?]}, \dots\dots, X_i^{[1]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_1^{[2]}\right) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Downarrow \\
Y_1^{[m_i]} = \left\{X_1^{[m_i]}, X_2^{[?]}, \dots, X_i^{[?]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_1^{[m_i]}\right)
\end{array}\right\} \sum_{r=1}^{m_i} G(Y_1^{[r]})$$

$$\left.\begin{array}{l}
Y_i^{[1]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots\dots, X_i^{[1]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_i^{[1]}\right) \\
Y_i^{[2]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots\dots, X_i^{[2]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_i^{[2]}\right) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Downarrow \\
Y_i^{[m_i]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots, X_i^{[m_i]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Longrightarrow G\left(Y_i^{[m_i]}\right)
\end{array}\right\} \sum_{r=1}^{m_i} G(Y_i^{[r]})$$

$$\left.\begin{array}{l}
Y_N^{[1]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots\dots, X_i^{[?]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[1]}\right\} \Longrightarrow G\left(Y_N^{[1]}\right) \\
Y_N^{[2]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots\dots, X_i^{[?]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[2]}\right\} \Longrightarrow G\left(Y_N^{[2]}\right) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Downarrow \\
Y_N^{[m_i]} = \left\{X_1^{[?]}, X_2^{[?]}, \dots, X_i^{[?]}, \dots\dots\dots, X_{N-1}^{[?]}, X_N^{[m_i]}\right\} \Longrightarrow G\left(Y_N^{[m_i]}\right)
\end{array}\right\} \sum_{r=1}^{m_i} G(Y_N^{[r]}) \tag{3.5}$$

It is a very hard to find the minimum of function $\sum_{r=1}^{m_i} G(Y_i^{[r]})^n$, because there are several possible local minima. For this reason, the objective function $\sum_{r=1}^{m_i} G(Y_i^{[r]})^n$ are converted into another topological space by building an easier function E and placing it in a new form known as a Homotopy Function.

$$J_i(q(x_i), T) = \sum_{r=1}^{m_i} G\left(Y_i^{[r]}\right) - T * E \quad , \qquad T \in [0, \infty) \tag{3.6}$$

Where $q(X_i^{[r]})$ is the probability distribution associated with agent i, who is taken as the uniform probability, defined as:

$$q\left(X_i^{[r]}\right) = \frac{1}{m_i} \qquad , i = 1,2,3, \ldots\ldots, m_i \tag{3.7}$$

As an illustration, the uniform probability distribution for agent i may look like that shown in figure 3.1 .for example, there are 5 strategies i.e $m_i = 5$. Then, each agent i will take uniform probability distribution $q\left(X_i^{[r]}\right) = \frac{1}{5}$.

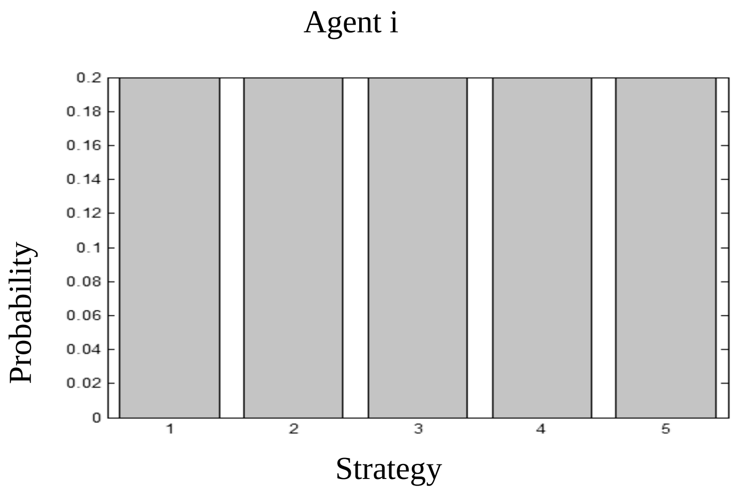


Figure 3.1: Uniform Probability Distribution $\boldsymbol{m_i = 5}$ of Agent i.

Each agent calculates the expected utility function $\sum_{r=1}^{m_i} E\left(G(Y_i^{[r]})\right)$ through using a joint product probability that is sampled randomly the probabilities from distributions of other agents as [4][13]:

$$\left.\begin{array}{l} G\left(Y_1^{[1]}\right)\ q\left(X_1^{[1]}\right) \prod_{(1)} q\left(X_{(1)}^{[?]}\right) = E\left(G\left(Y_1^{[1]}\right)\right) \\ \vdots \\ G\left(Y_1^{[r]}\right)\ q\left(X_1^{[r]}\right) \prod_{(1)} q\left(X_{(1)}^{[?]}\right) = E\left(G\left(Y_1^{[r]}\right)\right) \\ \vdots \\ G\left(Y_1^{[m_1]}\right)\ q\left(X_1^{[m_1]}\right) \prod_{(1)} q\left(X_{(1)}^{[?]}\right) = E\left(G\left(Y_1^{[m_1]}\right)\right) \\ \downarrow \end{array}\right\} \quad \begin{array}{c} \sum_{r=1}^{m_1} E\left(G\left(Y_1^{[m_1]}\right)\right) \\ \downarrow \end{array}$$

$$\left.\begin{aligned} G\left(Y_N^{[1]}\right)\, q\left(X_N^{[1]}\right) \prod_{(N)} q\left(X_{(N)}^{[?]}\right) &= E\left(G\left(Y_N^{[1]}\right)\right) \\ &\vdots \\ G\left(Y_N^{[r]}\right)\, q\left(X_N^{[r]}\right) \prod_{(N)} q\left(X_{(N)}^{[?]}\right) &= E\left(G\left(Y_N^{[r]}\right)\right) \\ &\vdots \\ G\left(Y_N^{[m_N]}\right)\, q\left(X_N^{[m_N]}\right) \prod_{(N)} q\left(X_{(N)}^{[?]}\right) &= E\left(G\left(Y_N^{[m_N]}\right)\right) \end{aligned}\right\} \sum_{r=N}^{m_N} E\left(G\left(Y_N^{[r]}\right)\right) \quad (3.8)$$

Now, we need to replace E used in the Homotopy function and put its place a convex function such as the Entropy Function [7][4].

$$S_i = -\sum_{r=1}^{m_i}\left[q\left(X_i^{[r]}\right)\log_2 q\left(X_i^{[r]}\right)\right] \quad (3.9)$$

Hence, each agent i minimizes its Homotopy function as:

$$J_i\left(q\left(X_i^{[r]}\right), T\right) = \sum_{r=1}^{m_i} E\left(G(Y_i^{[r]})\right) - T * \; S_i$$

$$= \sum_{r=1}^{m_i} E\left(G(Y_i^{[r]})\right) - T * \left(-\sum_{r=1}^{m_i}\left[q\left(X_i^{[r]}\right)\log_2 q\left(X_i^{[r]}\right)\right]\right) \quad (3.10)$$

where $T \in [0, \infty)$

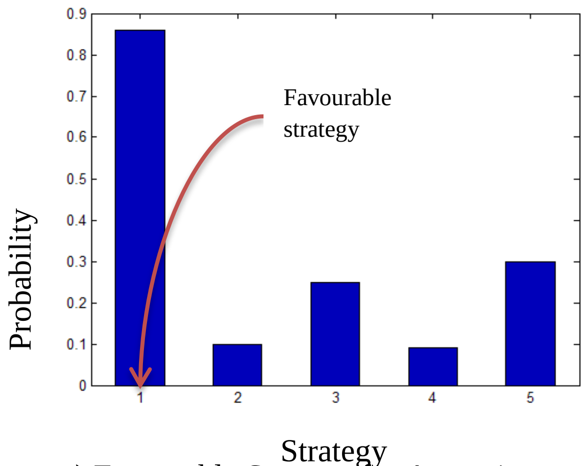
A suitable optimisation technique is used to find the minimization of Homotopy function such as Nearest Newton Descent Scheme (NNDS), Borden-Flectcher-Goldfarb-Shanno (BFGS) and Deterministic Annealing (DA) (Kulkarni et al 2015)[7][13]. This thesis introduces a further intensification scheme using the PC algorithm and based on results generated by Nearest Newton Descent Scheme (NNDS).

$$q\left(X_i^{[r]}\right) \leftarrow q\left(X_i^{[r]}\right) - \propto_{step} * \, q\left(X_i^{[r]}\right) * K_{r\ update} \quad (3.11)$$

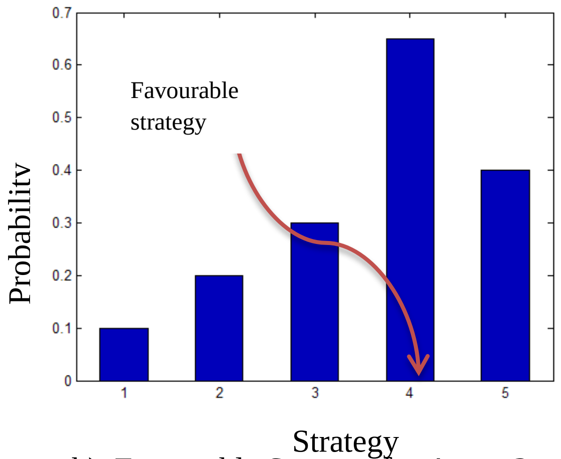
Where $K_{r\ update} = \frac{Contribution\ of\ agent\ i^r}{T} + \; s_i(q) + \ln\left(q\left(X_i^{[r]}\right)\right) \quad (3.12)$

And $Contribution\ of\ agent\ i^r = E\left(G\left(Y_i^{[r]}\right)\right)_n - \left(\sum_{r=1}^{m_i} E\left(G\left(Y_i^{[r]}\right)\right)\right)_n \quad (3.13)$

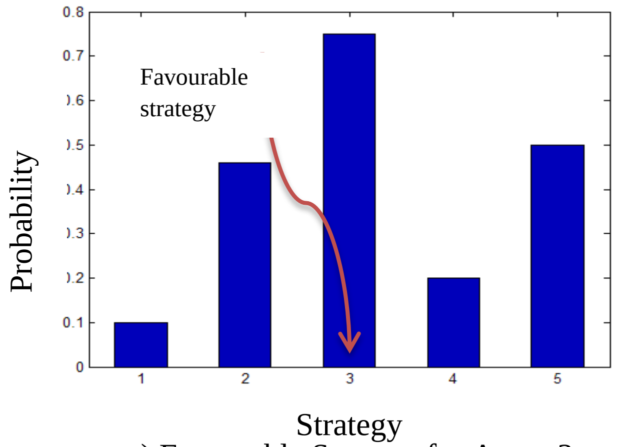
Where $\propto_{step}$ is constant which takes a value $\in (0, 1]$and T is Boltzamann's temperature, which starts from $\gg 0$ $or\ T = T_{intial}\ or\ T \to \infty$, so K is a number of iterations and $s_i(q)$ is the Entropy Function of agent i. After that, each agent i find the favourable strategy $X_i^{[fav]}$ by the highest prbability value over its distributin [13]. For example, there are 5 strategies for 3 agents as demonstrated in figure 3.2.



a) Favourable Strategy for Agent 1



b). Favourable Strategy for Agent 2



c) Favourable Strategy for Agent 3

Figure 3.2: Probability Distribution of Agent, i=1,2,3.

All agents compute the objective function $(G(Y^{fav})^n$ where Y_{fav} is given by $Y^{fav} = \{X_1^{fav,n}, X_2^{fav,n}, .., X_{N-1}^{fav,n}, X_N^{fav,n}\}$. Actually, there are some criteria to terminate the algorithm of probability collectives either:

- If temperature $T \rightarrow 0$.
- If $\| \ G(Y^{fav})_n - G(Y^{fav})_{n-1} \| \leq \ \varepsilon$ where $\varepsilon \ > 0$.

For each iteration, the PC algorithm updates the boundaries of variables Ω and Boltzmann's temperature as follows:

$X_i^L(n+1) = (1-\lambda) * X_i^{fav}$, i=1… N (3.14)

$X_i^H(n+1) = (1+\lambda) * X_i^{fav}$, i = 1,………, N (3.15)

$T_{n+1} = (1-\alpha_T) * T_n$ (3.16)

Where $0 < \lambda < 1$ is the range factor and $0 < \alpha_T < 1$ is the cooling rate. The algorithm PC continues until one of mentioned criteria above is satisfied.

3.3 Implantation of PC Algorithm

The probability collectives algorithm to solve problems is as follows:

- **Initialize**
- Set the parameters $(T, \alpha_S, \alpha_T, \lambda)$ and convergence criteria ε.
- Allocate the starting probabilities for each agent to uniform over its values as Eq. (3.7).
- Set the number of generations $N \leftarrow 0$.
- **Repeat**
- Form a set of combined strategies $Y_i^{[m_i]}$ for each agent i as Eq. (3.3).
- Evaluate the objective function for the set of combined strategies $Y_i^{[m_i]}$ as Eq. (3.4).
- For each agent i computes expected objective function using Eq. (3.8).
- Minimise the Homotopy FunctionJ_i as shown in Eq. (3.10).
- Compute the contribution of $X_i^{[r]}$using Eq. (3.13).
- Update the probabilities of all the strategies for every agent, I as Eq. (3.11).
- Update the strategy boundaries using Eq. (3.14) and (3.15).
- Update Boltzmann’s temperature as shown in Eq. (3.16).
- Define the favourable strategy through determining the maximum probability value for each variable as shown in Figure 3.2.

- Compute the utility function $(G(Y^{fav})$ with this set of agents.
- Keep the favourable strategy for every value.
- $N = N + 1$.
- **Until** the convergence criteria are satisfied.
- Accept final value.
- Stop

The flowchart description of PC algorithm is shown in Figure 3.3.

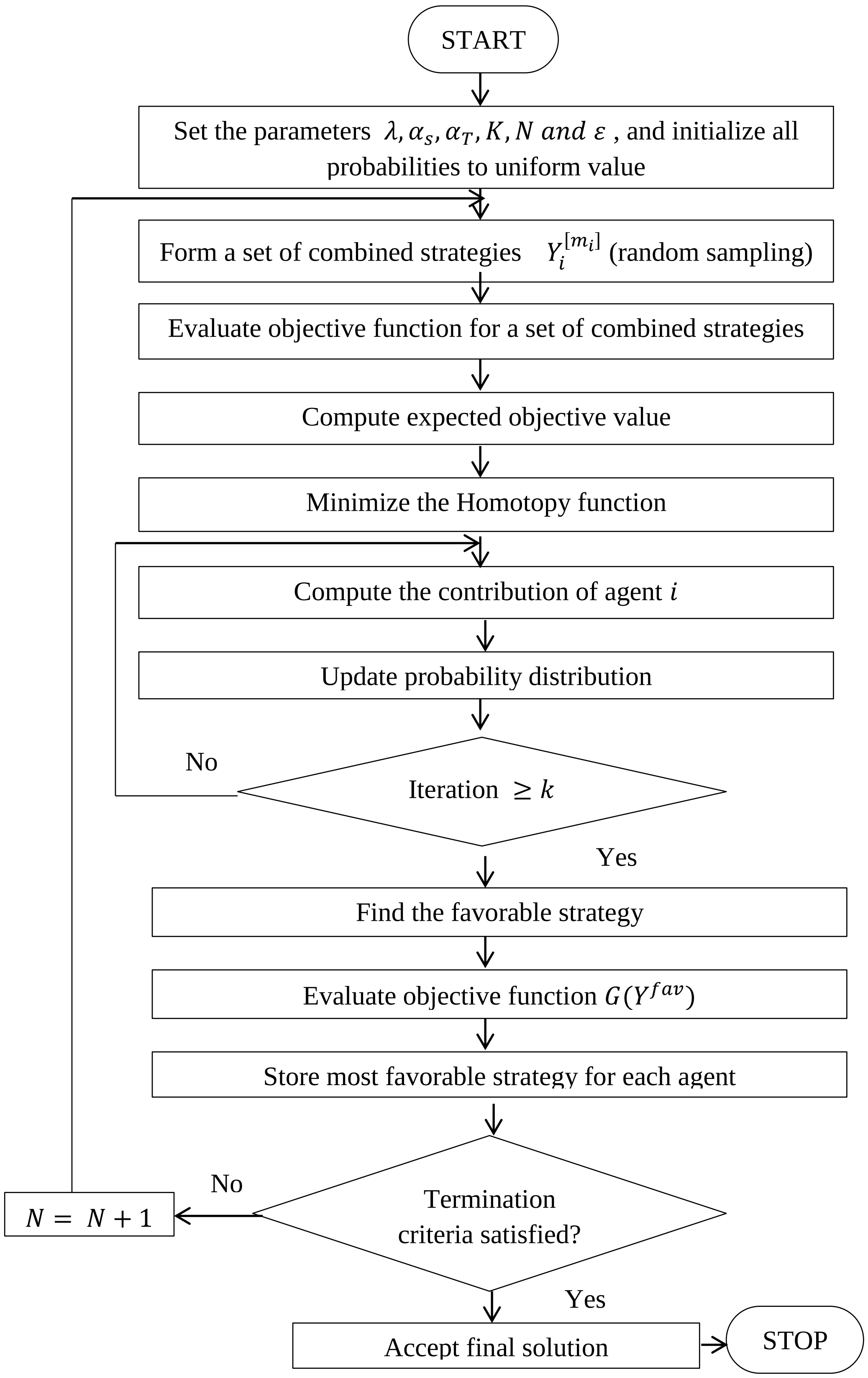

START
Set the parameters $\lambda, \alpha_S, \alpha_T, K, N\ and\ \varepsilon$, and initialize all probabilities to uniform value
Form a set of combined strategies $Y_i^{[m_i]}$ (random sampling)
Evaluate objective function for a set of combined strategies
Compute expected objective value
Minimize the Homotopy function
Compute the contribution of agent i
Update probability distribution
No
Iteration $\geq k$
Yes
Find the favorable strategy
Evaluate objective function $G(Y^{fav})$
Store most favorable strategy for each agent
No
$N = N + 1$
Termination criteria satisfied?
Yes
Accept final solution
STOP

Figure 3.3: Flowchart of PC Algorithm

Chapter 4

METHODOLOGY

In this thesis, we applied the probability collectives algorithm to solve four different problems such as 23 Numerical benchmark functions, El Farol bar, constraint satisfaction (N-Queens), and Repeated games problems.

4.1 Application of Probability Collectives to Numerical Benchmark Problems

In this thesis, we used the PC algorithm described in chapter 3 to solve 23 Numerical benchmark problems, which are described in chapter 2. In the initialization of parameters, we set the number of runs (Run=20) and the number of iteration (Max_it) is a different for each problem as well as the number of agent (N). We also set the number strategies ($M_i = 20$) for each agent and parameters including ($T = 10^5, \lambda = 0.9, \alpha_S = 0.098, \alpha_T = 0.98, K = 1000, \varepsilon = 0.00001$). After that, we proceeded with the following steps:

- Allocate uniform probability value ($st_{i,j} = \frac{1}{20}$) for each agent.
- Each variable (agent) is randomly chosen value from $[x^l, x^h]$ i.e. [-100,100] for sphere function.
- Form a set of combined strategies $Y_i^{[m_i]}$.
- Evaluate objective function for each problem as shown in table 2.1 as well as calculate expected and Homotopy function.
- for each k of iterations updates each probability values.

- find the highest of probability to find the best value for each agent, and then evaluate objective function.

- IF the difference between the best solution and the current solution is less than ε, then the final solution is accepted.

- Else updates the boundaries of variables Ω and Boltzmann's temperature and $N = N + 1$.

The flowchart of Numerical benchmark problems is the same, which is described in chapter 3.

4.2 Application of PC to El Farol Bar Problem

In this problem, we applied the PC algorithm as follows:

a- Initial parameters ($T, \lambda, \alpha_S, \alpha_T, K, M_i, Runs$), as the same first problem as well as the parameters of El Farol bar problem is as:

- M_z is the size of memory where M_z =8.
- N is the number of agents where $N = 100$.
- W is the number of weeks where $W = 1000$.
- The size of history attendance equal M_z * 2.
- The size of strategy equal M_z + 1.

b- Initialize the current history of attendance randomly among (0:100) such as:

His_at= [10,20,50,15,60,100,12,0,18,80,50,59,60,40,25,32] subject to His_at is integer numbers .

c- For each week, find the number of attendance as:

- Initial the strategy to each agent among (-1,1)as shown:

$ST_1 = [c, w_1, w_2, \ldots\ldots w_{M_Z}]$

$\Downarrow$

$ST_S = [c, w_1, w_{2,} \ldots\ldots, w_{M_Z}]$, where c is constant and w is a weight.

- Calculate prediction for each agent using Eq. (2.1).

- Find the best strategy by computing the strategy score for each strategy using Eq. (2.2). Hence, the low score is good, a high score is bad. Mathematical example to show how to find the best strategy and calculate a score [25].

Each strategy based on M_z=2 and M=2.

History of agent 1

His_at_1	His_at_2	His_at_3	His_at_4
30	20	5	90

Strategy 1

C	W_1	W_2
0	0.5	0.5

Strategy 2

C	W_1	W_2
0.5	-0.1	0.6

To compute the score:

p_0 = Predict His_at_1 using His_at_2, His_at_3, $p_0 = C + w_1 * His\_at_2 + w_2 * His\_at_3$.

p_1 = Predict His_at_2 using His_at_3, His_at_4, $p_1 = C + w_1 * His\_at_3 + w_2 * His\_at_4$.

$error = |His\_at_1 - p_0| + |His\_at_2 - p_1|$

Strategy 1 score:

$p_0 = 0 + 20 * 0.5 + 5 * 0.5$

=12.5

$p_1 = 0 + 5 * 0.5 + 90 * 0.5$

=47.5

$error = |30 - 12.5| + 20 - 47.5|$

$= 45$

Strategy 2 score:

$p_0 = 0.5 + 20 * -0.1 + 5 * 0.6$

=3.3

$p_1 = 0.5 + 5 * -0.1 + 90 * 0.6$

=55

$error = |30 - 3.3| + 20 - 55|$

=61.7

Strategy 1 is the best because the score is lower.

- Repeat this calculation for all agents then find minimum score using the PC algorithm as the same which discussed in chapter 3.
- Check the prediction if (p <= 60) then $sum\_at = sum\_at + 1$. This means, this agent comes to bar otherwise stays at home.
- Update the history attendance.
- Update temperature T.
- Repeat until finding a prediction for all week.
- Show results

Figure (4.1) shows the flowchart of the PC algorithm to the El Farol bar problem.

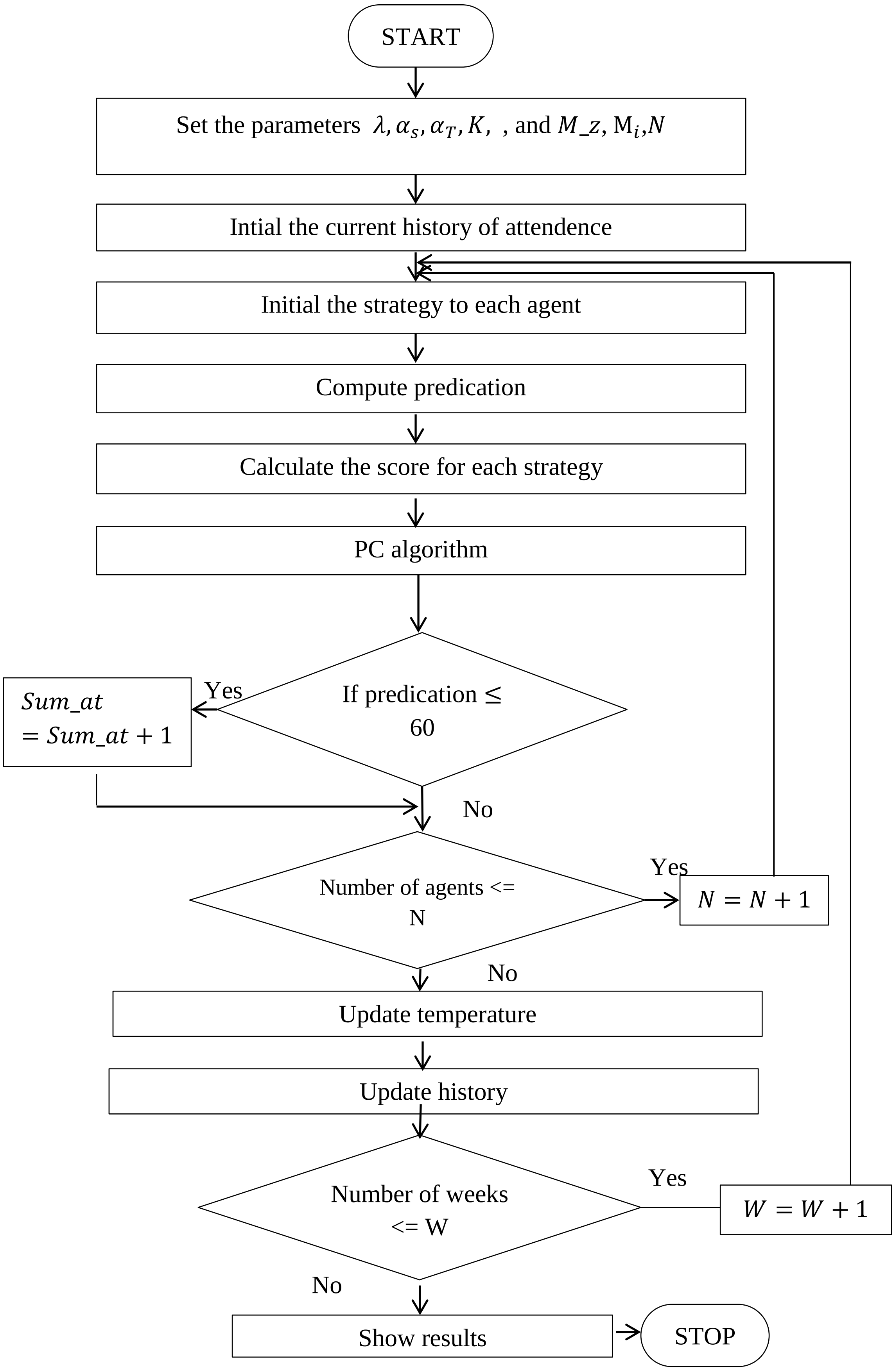


Figure 4.1: Flowchart PC Algorithm for El Farol Bar Problem

4.3 Application of PC to N-queens Problem

In this thesis, we applied this problem addressed by PC to Distributed Constraint Satisfaction Problems (DCSP). We can formalise the algorithm as follows:

- Initialize the parameters of PC algorithm ($T = 100, \lambda = 0.9, \alpha_{\mathrm{S}} = 0.098, \alpha_{\mathrm{T}} = 0.9, \mathrm{K} = 150, \mathrm{M}_i = 10, Runs = 20$), and ($NQ = 40$ is a number of queens, $N = 10$ is a number of population and $NAction =$ is a number of actions.
- Create actions list.
- Initialize a set of agents where each agent can take a value from domain $=\{1,2,\ldots.,NQ\}$ where initialization is done using a random permutation such as X=[1,3,4,2].
- Apply actions and evaluate the objective function for each agent.
- IF all constraint is not satisfied repeat pervious step otherwise go to next step.
- Find the best solution using the PC algorithm.
- Repeat until the global minimum is reached, then accept the optimal solution.

Figure 4.2 shows the flowchart of PC algorithm to N-queens problem.

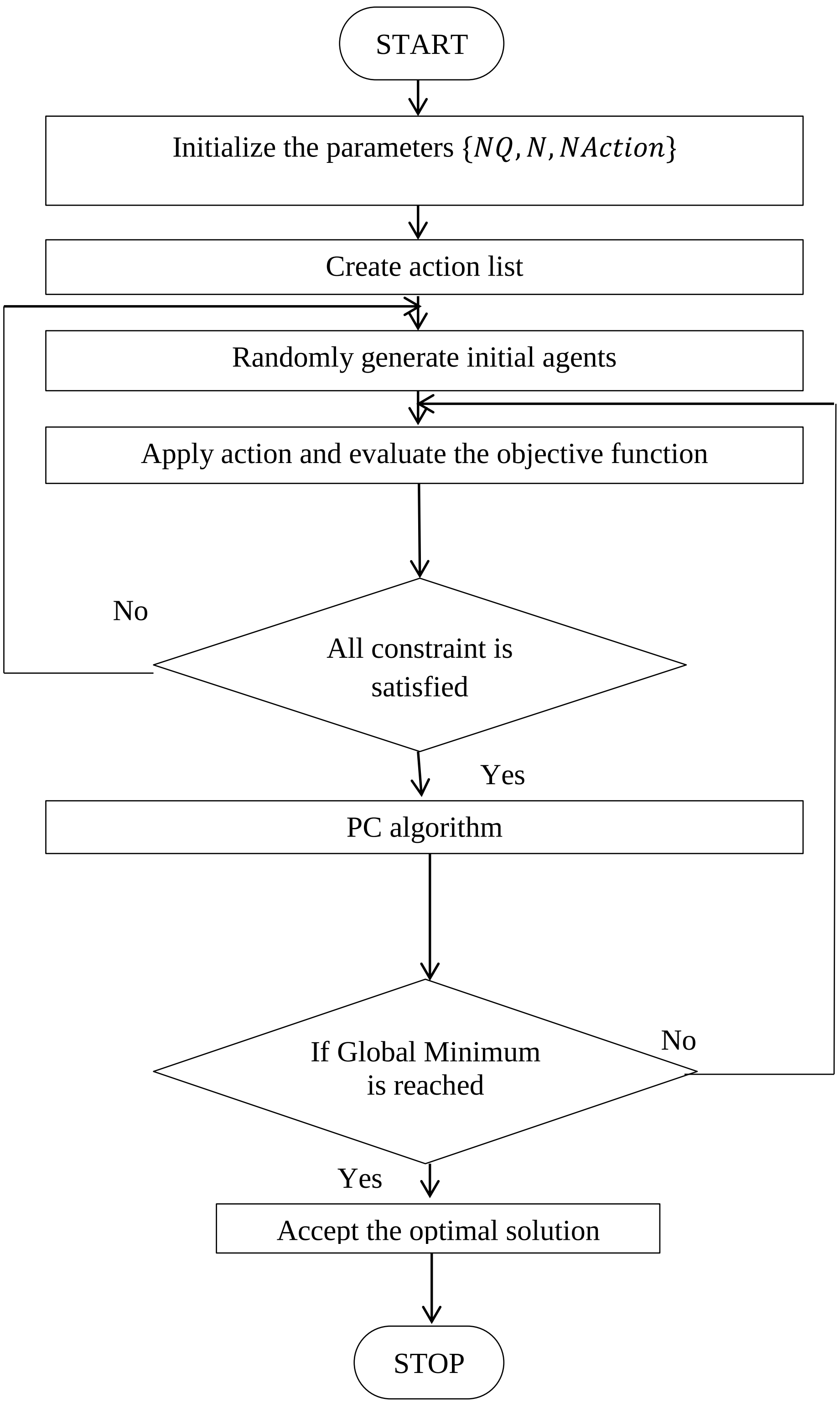

START
Initialize the parameters $\{NQ, N, NAction\}$
Create action list
Randomly generate initial agents
Apply action and evaluate the objective function
No
All constraint is satisfied
Yes
PC algorithm
If Global Minimum is reached
No
Yes
Accept the optimal solution
STOP

Figure 4.2: Flowchart PC Algorithm for N-queens Problem

4.4 Repeated Games

We applied the PC algorithm to the study of evolutionary game theory. In this thesis, we used different problems such as (Prisoner's dilemma, Stag hunt, Battle of sexes game, Choose sides) to simulate one shot (repeated) games between two players. Each X is considered as an agent in a D-dimensional space and each element of agent can take the binary value of 24-bit length. Also each bit will represent an action (0 for defect, 1 for cooperate) .all agents have a D-dimensional, which are in range $[0, 2^{lenght}]$. More details of the algorithm are as follows:

Step1: initial parameters of PC algorithm ($T, \lambda =, \alpha_S, \alpha_T, K, M_i, N = 10, maxit$).

Step2: allocate uniform probability value for each agent as$st_{i,j} = \frac{1}{10}$.

Step3: initial $X_i^{[m_i]}$ random such as $X_1 = \left\{1010101_1^{[1]}, 1010111_1^{[2]}, \dots \dots \dots, X_1^{[m_i]}\right\}$.

Step4: find a set of solution $Y_i^{[m_i]}$ then evaluate fitness function. For example, calculate objective function for Prisoner's dilemma as

IF (C&C) then fitness= fitness+3

IF (C&D) then fitness= fitness+0

IF (D&C) then fitness= fitness+5

IF (D&D) then fitness= fitness+1

Where C means a cooperate, D means a defect.

Step5: find expected function and homotopy function.

Step6: update probabilities value for k of iteration.

Step7: find the maximum probability.

Step8: evaluate objective function, calculate average payoffs, and update T, D

Step9: repeat until the number of iteration $\geq maxit$**,** and then show the results.

The flowchart of repeated games is the same which we described in chapter 3.

Chapter 5

EXPERIMENTAL RESULTS

The probability collectives is implemented to solve four problems using Matlab2013b in Windows 7 operating systems and run on a personal computer with Intel core(TM) i3 2.10 GHz CPU and 4.00GB of RAM. First, we focus on 23 benchmark functions, which the same functions used in (Huang et al.2005). Moreover, we compared the convergence rate of PC algorithm over 20 runs for unimodal functions $F_1 - F_7$ except for F_6 which is a discountouns function as well as compared highly multimodal functions $F_8 - F_{13}$ and basic multimodal functions $F_{14} - F_{23}$. Second, the simulation of el Farol bar problem is done over 1000 weeks. We will show the mean and standard deviation of attendance over 20 runs. Third, we implemented the PC algorithm of N-queens problem for a various sizes N=(8,100,150,180) and show the best solution based on time and a number of iterations. Fourth, different related games are solved using PC such as (Prisoner's dilemma, Stag hunt, Battle of sexes game, Choose sides). In these games, we did simulation for 10 agents over 500 times and will show the comparison among four games, which reach the best fitness.

5.1 Results of Benchmark Problem

The first experiment is carried on benchmark functions $F_1 - F_7$, which have 30 variables (agents). Each agent has 20 strategies, starting from uniform probability $\frac{1}{20}$ and random initial values. So, all functions are iterated 2000 times until convergence is reached to ε =0.000001. We discussed only Sphere function while you can see the results of other functions $F_2 - F_7$ in the following figures 5.2 and 5.3. In figure 5.1(a), the result of function F_1 reached the best minimum solution about 1.0797e-07 at 1 run. Because, the probability of some agents are close to one such as agents (3,4,7,8,11,15,20,21,26,29) as indicated in figure 5.1 (b). However, the worst solution of Sphere function is 1.975e-05 at 13 run.

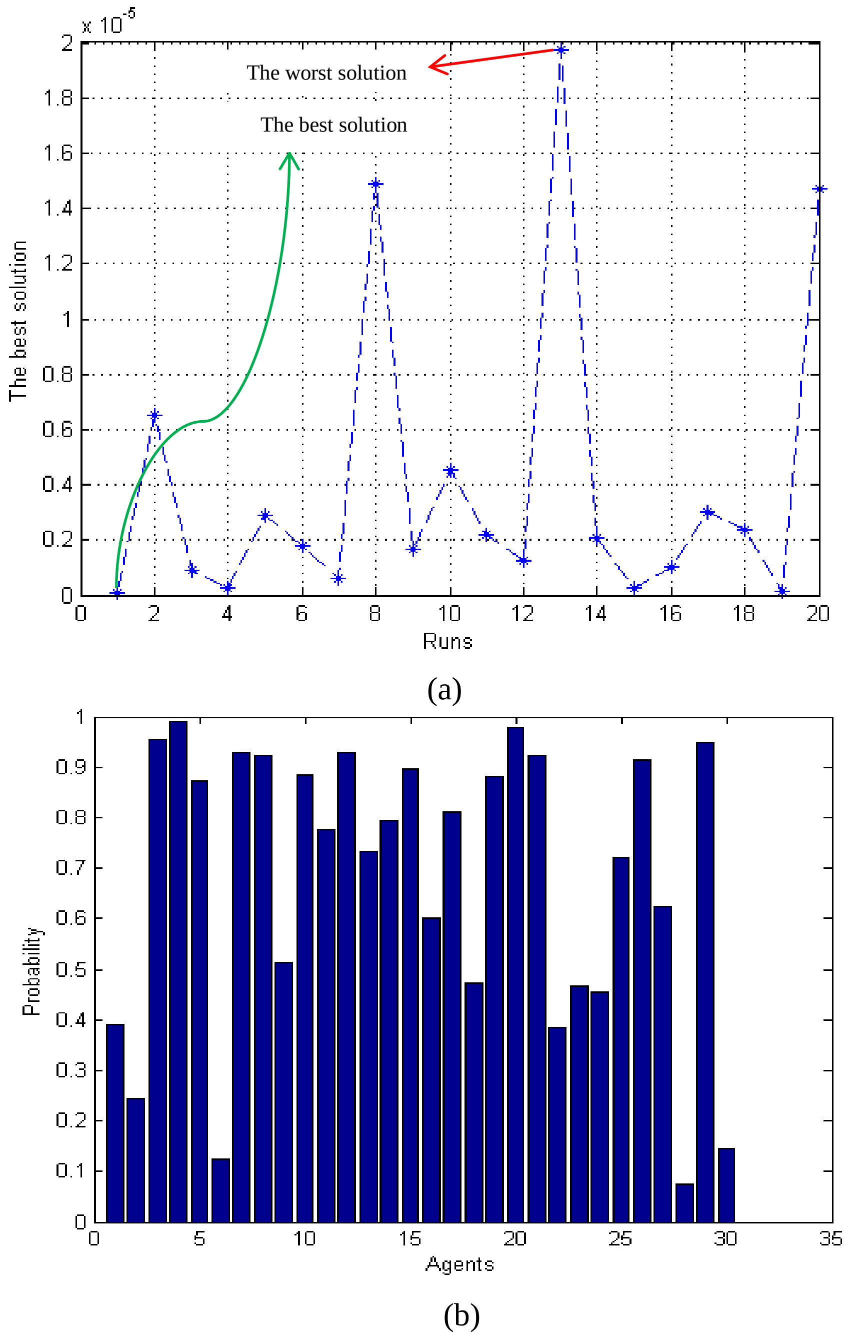
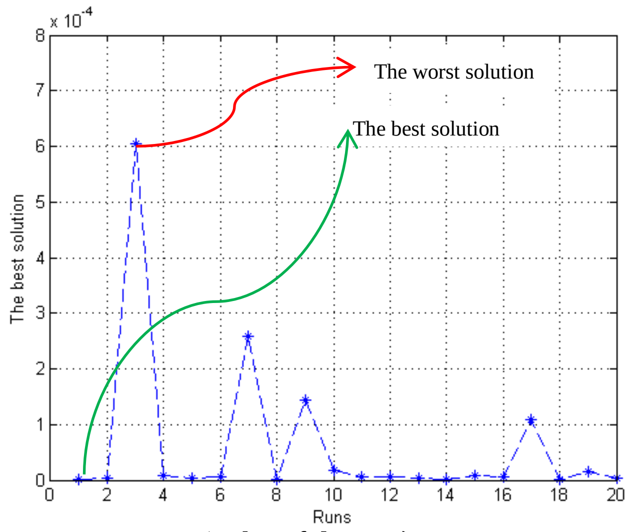
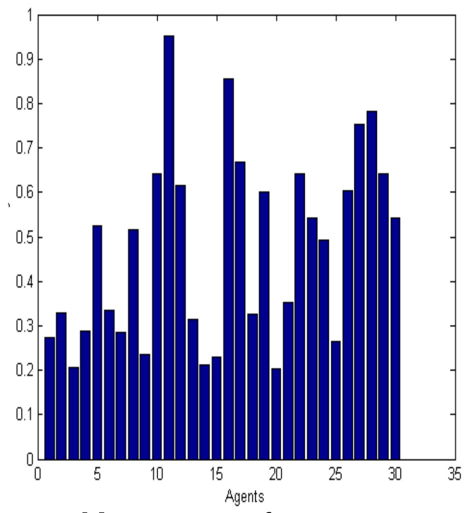


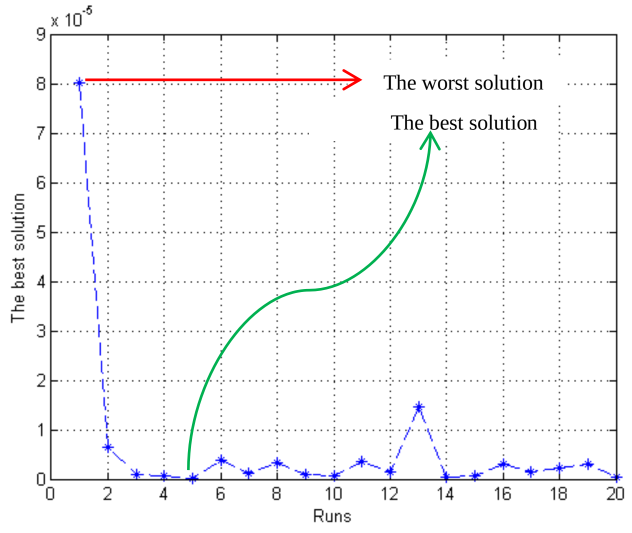
Figure 5.1: Performance of The PC Algorithm on Sphere Function. (a) Shows The Best Result Over 20 Runs, and (b) Shows Favourable Strategy of Agents at 1 Run.



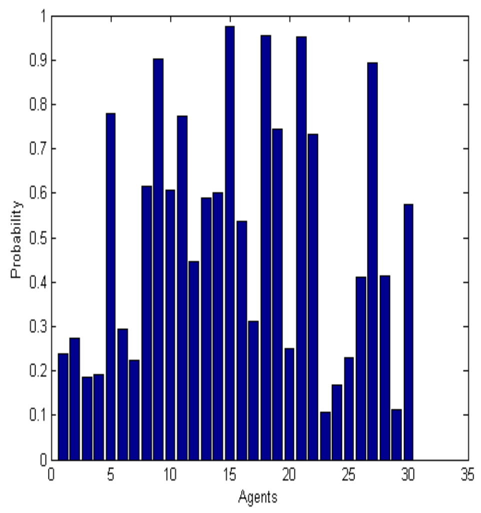
F_2 (Schwefel 2.22)



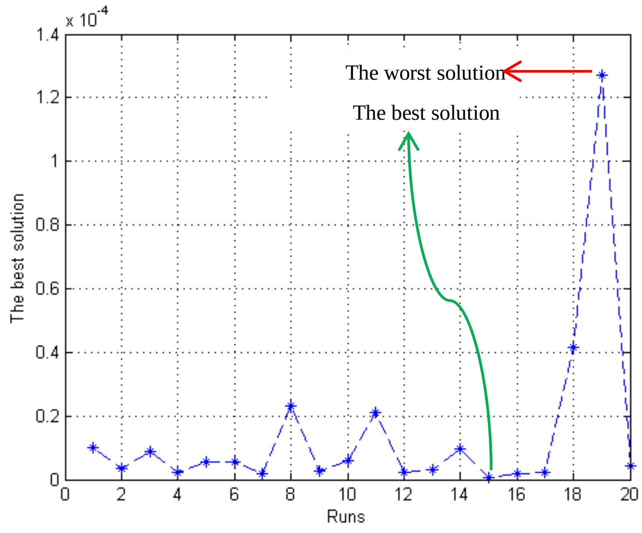
Favorable Strategy of 30 agents at 1

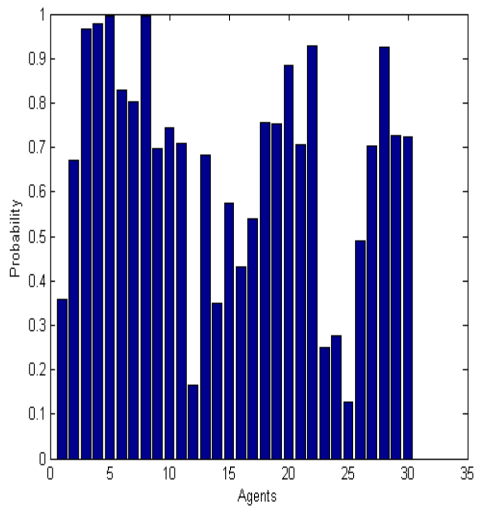


F_3 (Schwefel 1.2)



favorable strategy of 30 agents at run 5

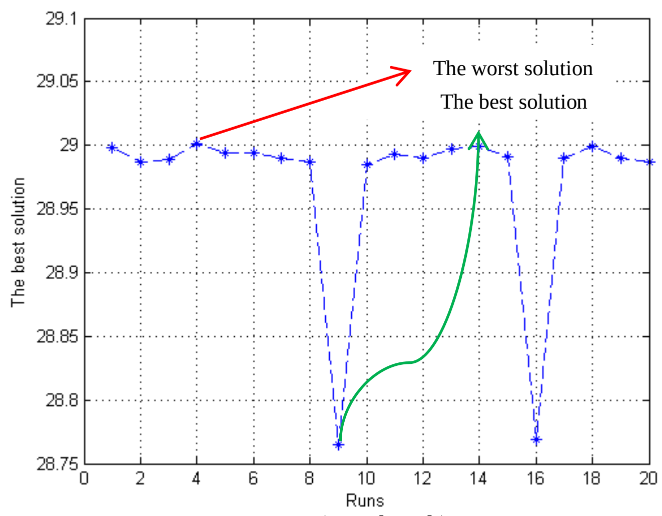




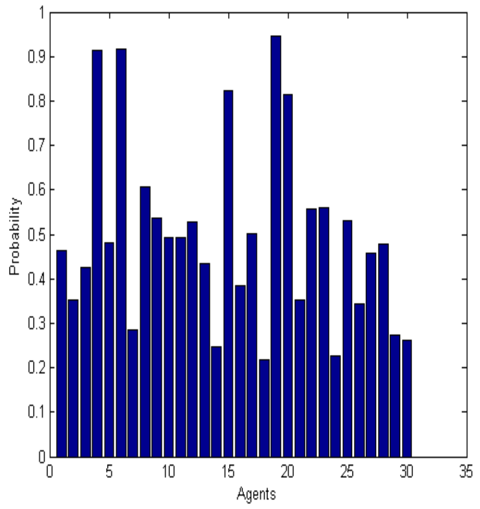
F_4 (Schwefel 2.21) Favorable Strategy of 30 agents at run 14

(a) Shows the best result over 20 runs (b) Shows favourable strategy of 30 agents

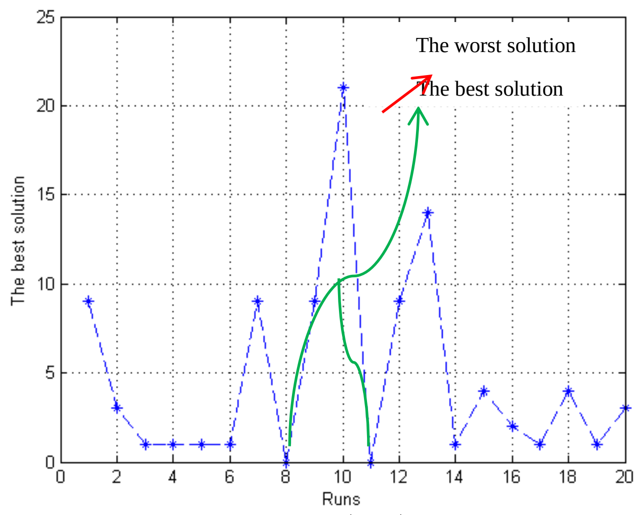
Figure 5.2: Performance of The PC Algorithm on Functions $F_2 - F_4$.



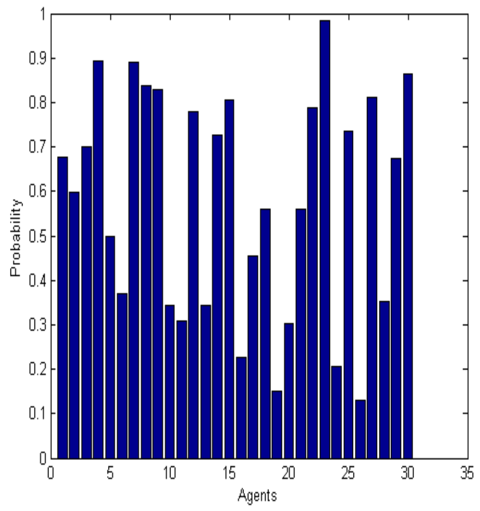
F_5(Rosebrock)



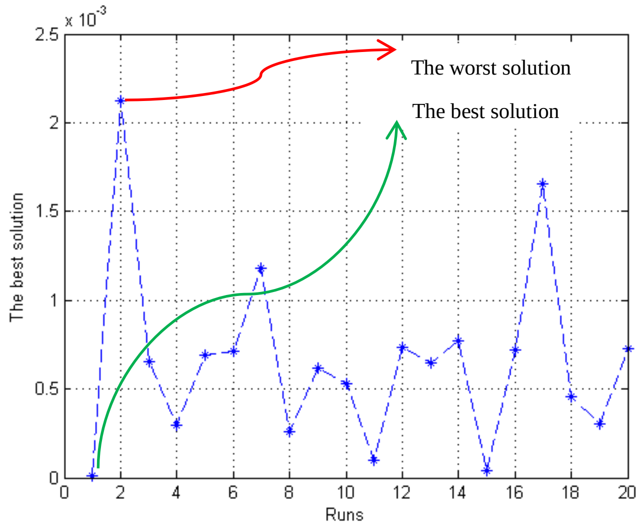
Favorable Strategy of 30 agents at run 9



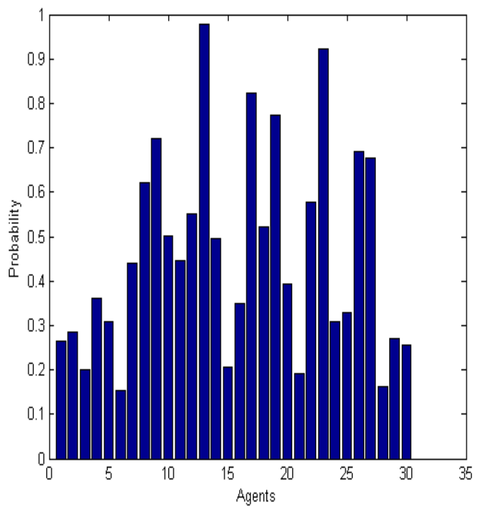
F_6(Step)



Favorable strategy of 30 agents at run 8



F_7 (Quartic)



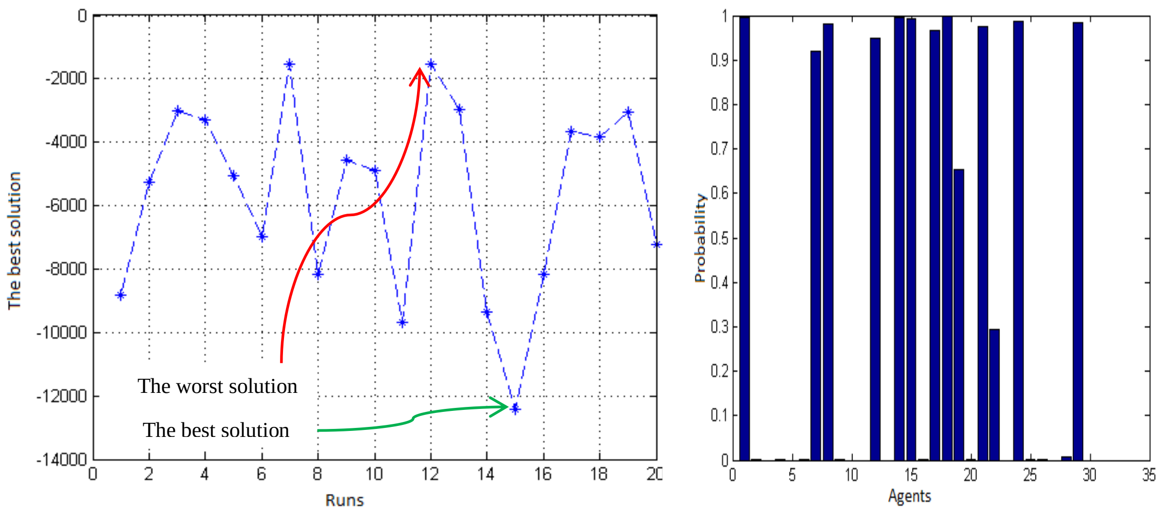
Favorable strategy of 30 agents at run 1

(a) Shows the best result over 20 runs (b) Shows favourable strategy of 30 agents

Figure 5.3: Performance of The PC Algorithm on Functions $F_5 - F_6$.

From Figures 5.1, 5.2 and 5.3, we can summarise the results of these functions where the best result is function 6 which converged to zero. However, the worst result is function 5 because the most its results are close to 29, while the results of other functions are similar approximately between $10^{-5} - 10^{-8}$.

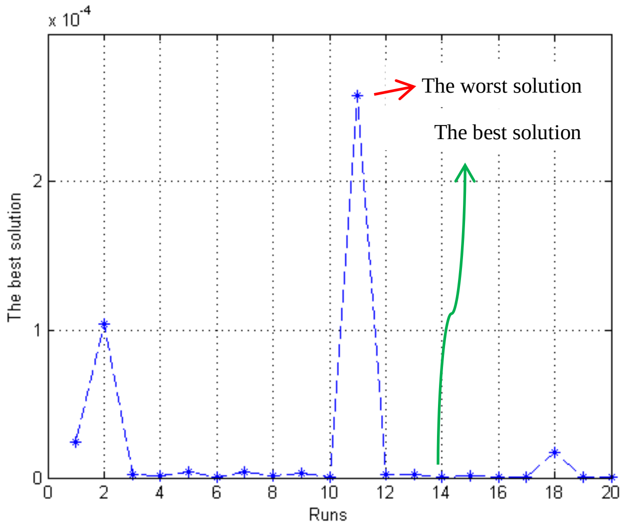
High-dimensional functions $F_8 - F_{13}$ are the second experiment that is implemented using the PC algorithm. They have the same dimension and also the number of strategies that are used in the first experiment. But the number of iterations is a different in around 1000 for all functions. In Figures 5.4, 5.5, and 5.6 (a), the results of functions $F_8 - F_{11}$ are the nearest to the global optimum such as (our result of $F_8 =$ -12435.8614, global optimum= -12569.5). In contrast, outcomes of functions $F_{12} - F_{13}$ gave us local solutions (F_{12} approximately 0.1, F_{13} about 0.3) which are far the optimum solutions (0).



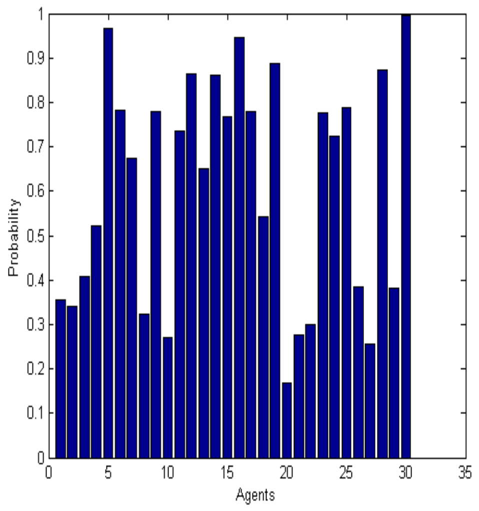
F_8 (Schwefel2.26) Favorable strategy of 30 agents at run 15

(a) Shows the best result over 20 runs (b) Shows favourable strategy of 30 agents

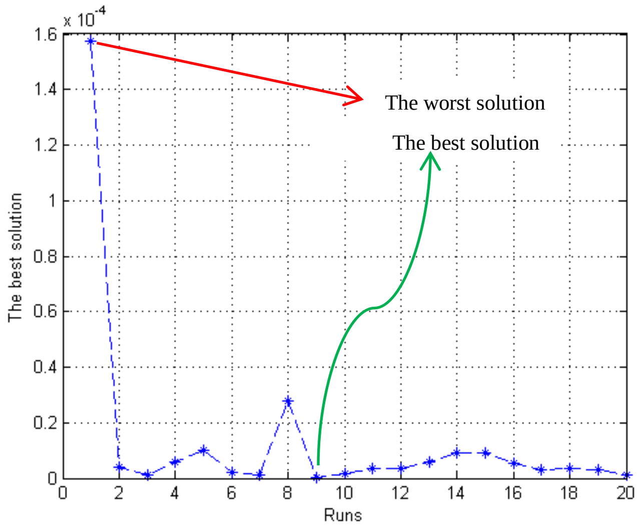
Figure 5.4: Performance of The PC Algorithm on Function F_8.



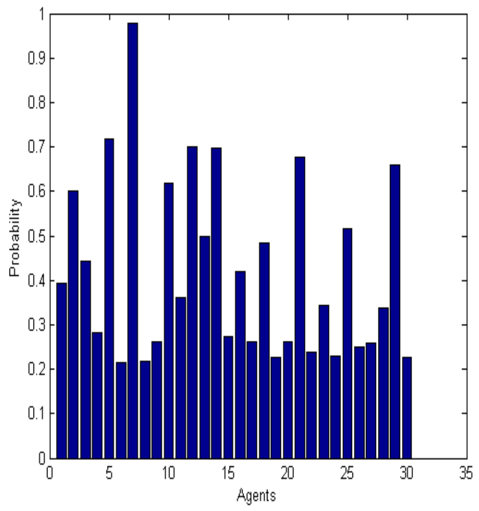
F_9 (Rastriging)



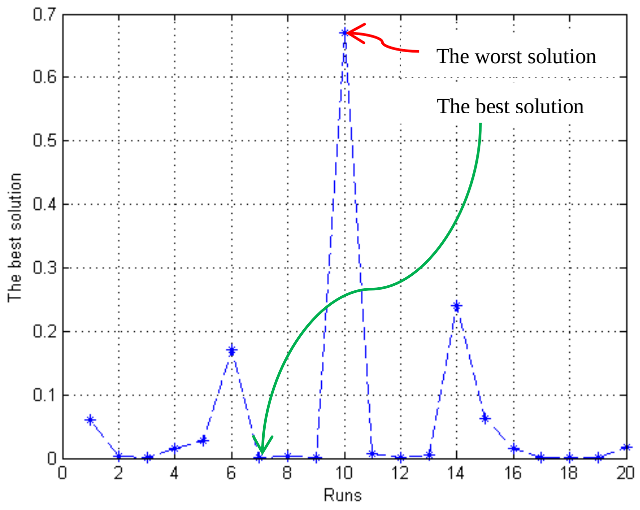
Favorable strategy of 30 agents at run 14



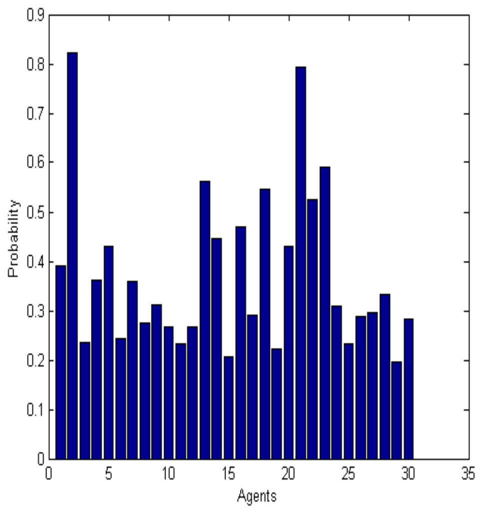
F_{10} (Ackley)



Favorable strategy of 30 agents at run 9



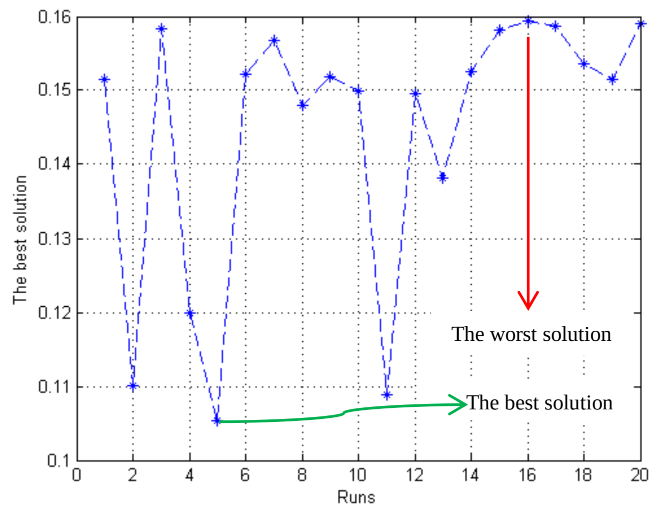
F_{11} (Griewank)



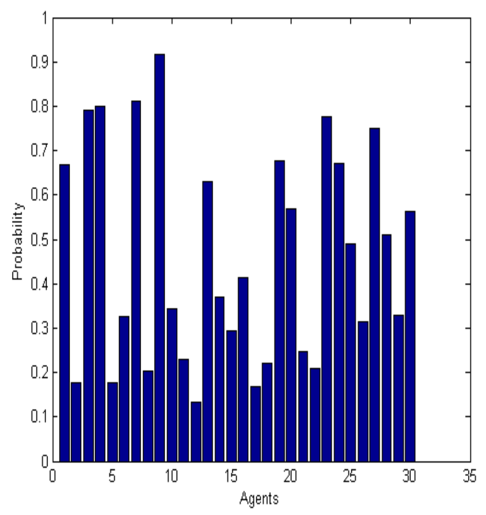
Favorable strategy of 30 agents at run 7

(a) Shows the best result over 20 runs (b) shows favourable strategy of 30 agents

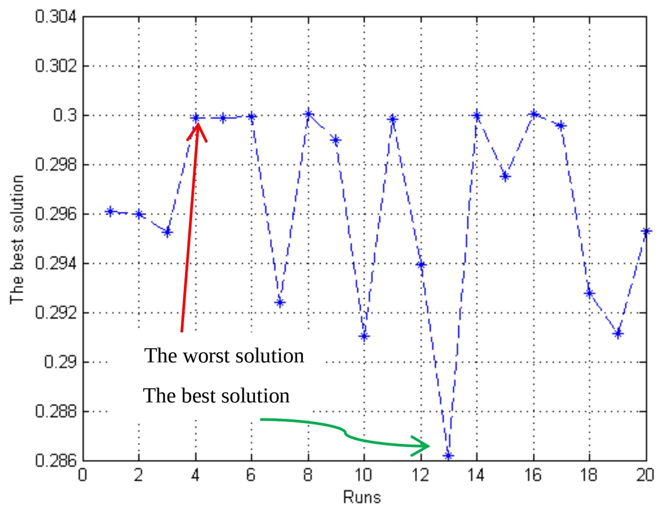
Figure 5.5: Performance of The PC Algorithm on Functions $F_9 - F_{11}$.



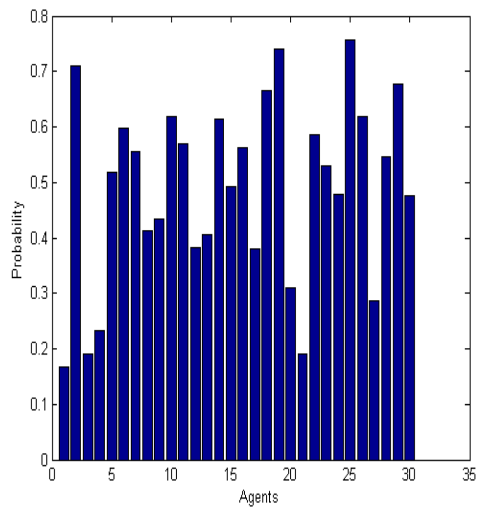
F_{12} (Levy)



Favorable strategy of 30 agents at run 5



F_{13} (Levy 8)

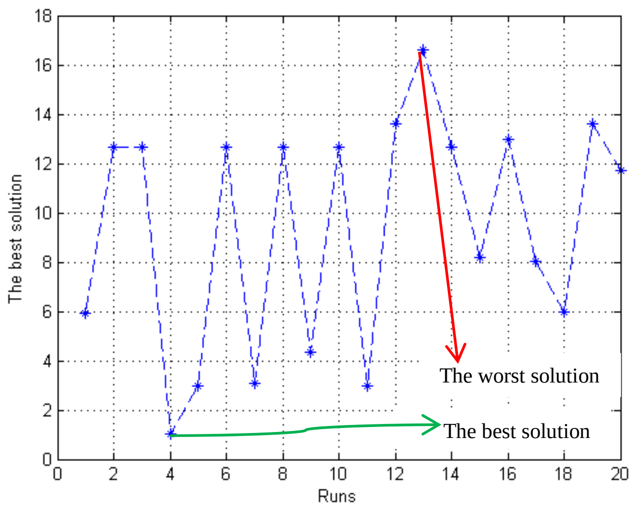


Favorable strategy of 30 agents at run 13

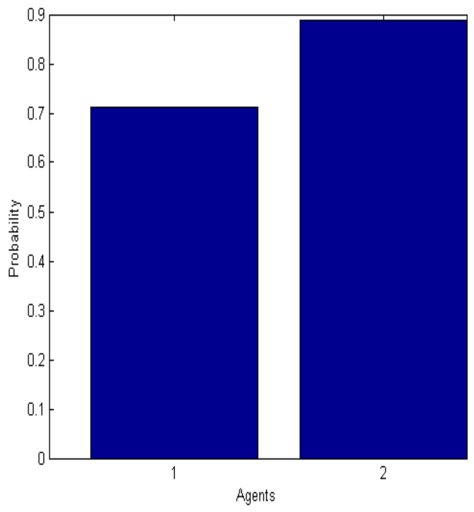
(a) Shows the best result over 20 runs (b) Shows favourable strategy of 30 agents

Figure 5.6: Performance of The PC Algorithm on Functions $F_{12} - F_{13}$.

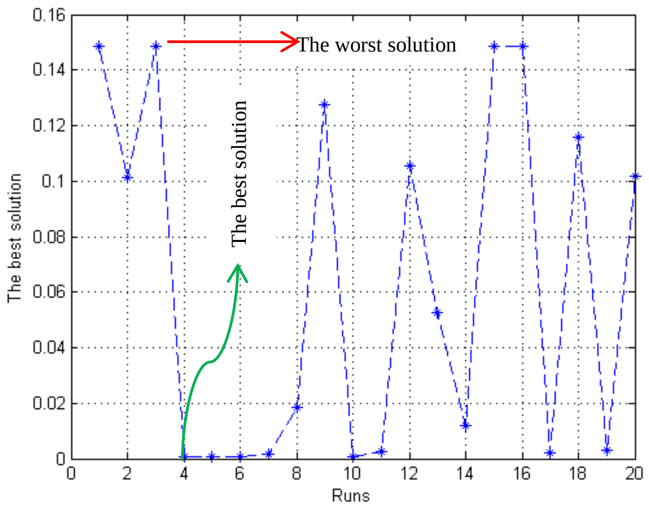
The third experiment is executed on low-dimensional functions $F_{14} - F_{23}$. Each function has a various dimension and the number of iteration. In figures 5.7, 5.8, 5.9, and 5.10, all functions $F_{14} - F_{23}$ are converged to the global solutions such as (our outcome of function F_{16}= -1.0316, the global minimum = -1.0316285).



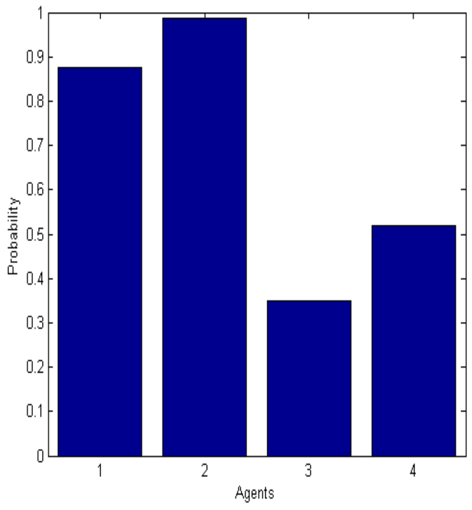
F_{14} (Shekel Foxholes)



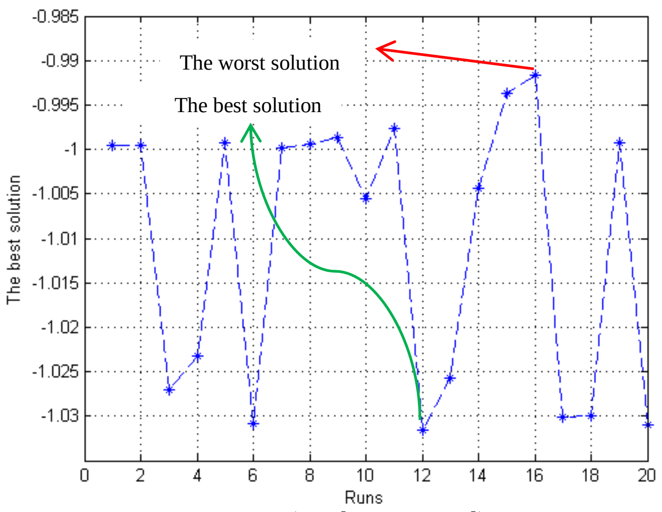
Favorable strategy of 2 agents at run 13



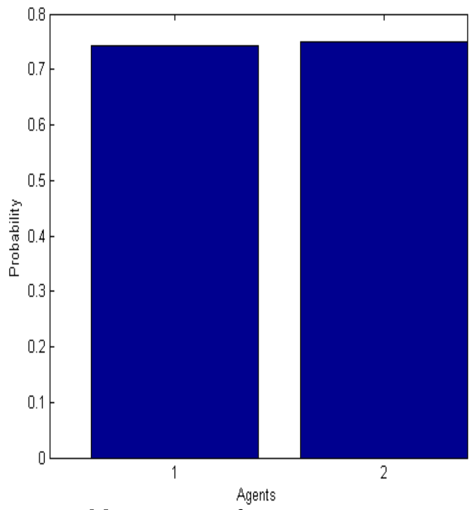
F_{15} (Kowalik)



Favorable strategy of 4 agents at run 4



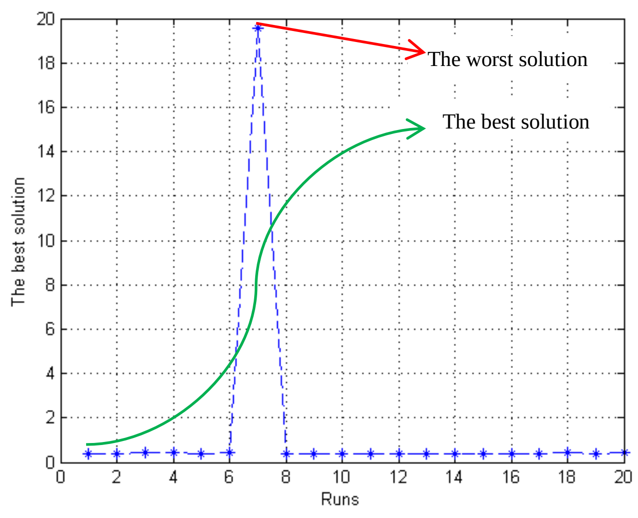
F_{16} (Six-hump Camel)



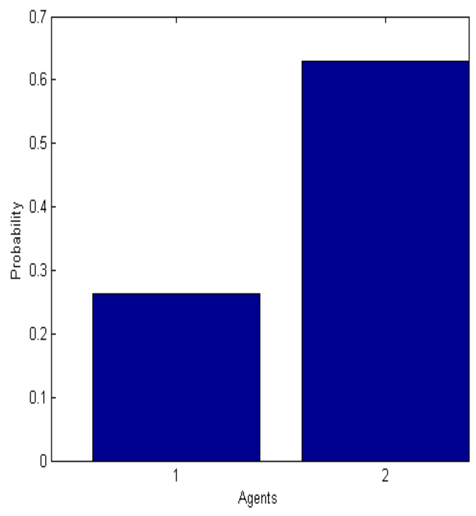
Favorable strategy of 2 agents at run 12

(a) Shows the best result over 20 runs (b) Shows favourable strategy of N agents

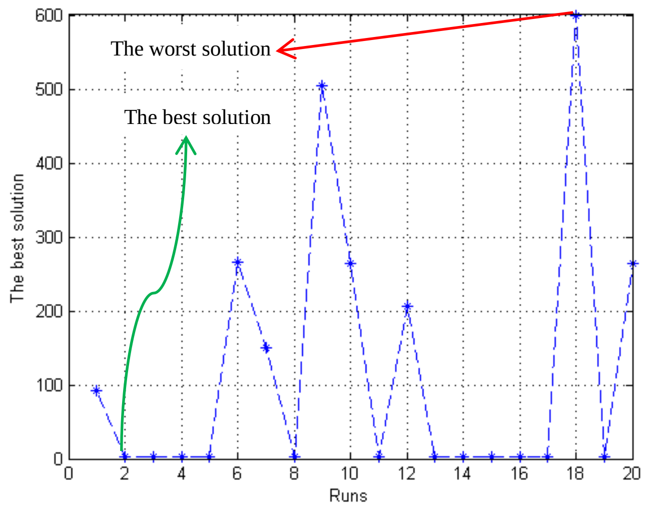
Figure 5.7: Performance of The PC Algorithm on Functions $F_{14} - F_{16}$.



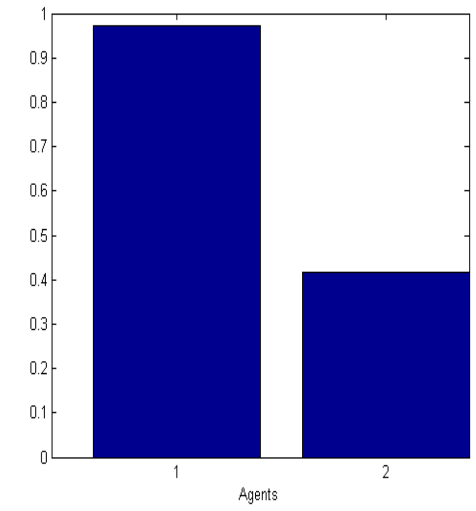
F_{17} (Branin)



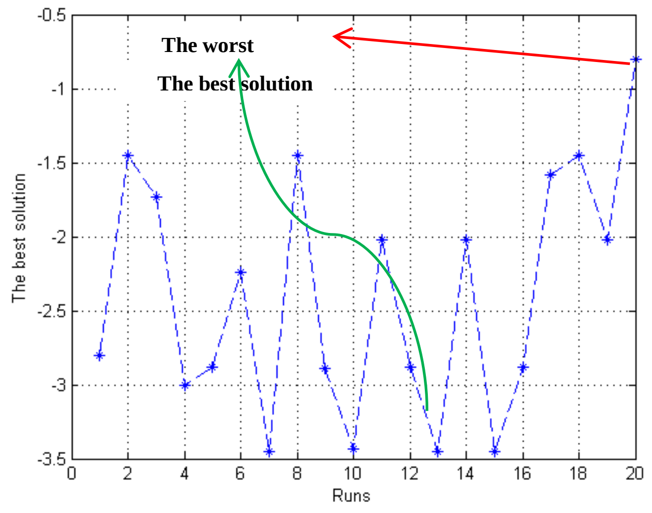
Favorable strategy of 2 agents at run 1



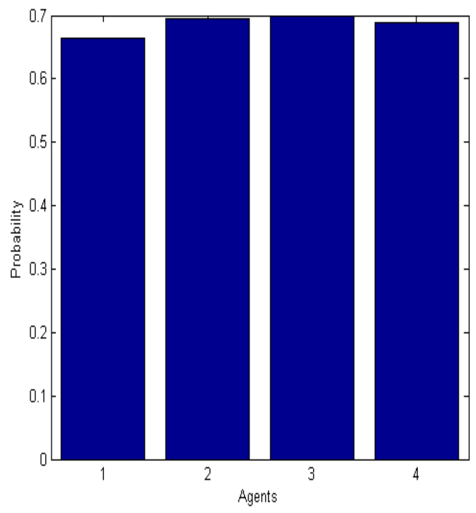
F_{18} (Goldstein-Price)



Favorable strategy of 2 agents at run 2



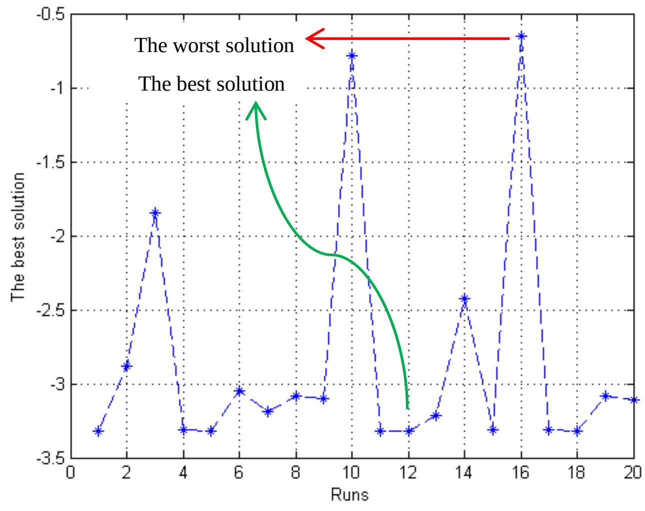
F_{19} (Hartman 4)



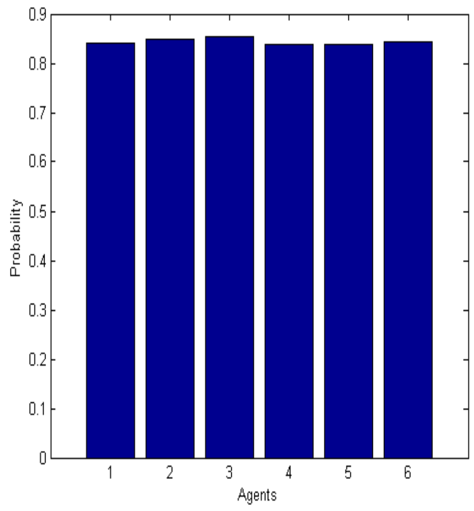
Favorable strategy of 4 agents at run 13

(a) Shows the best result over 20 runs (b) Shows favourable strategy of N agents

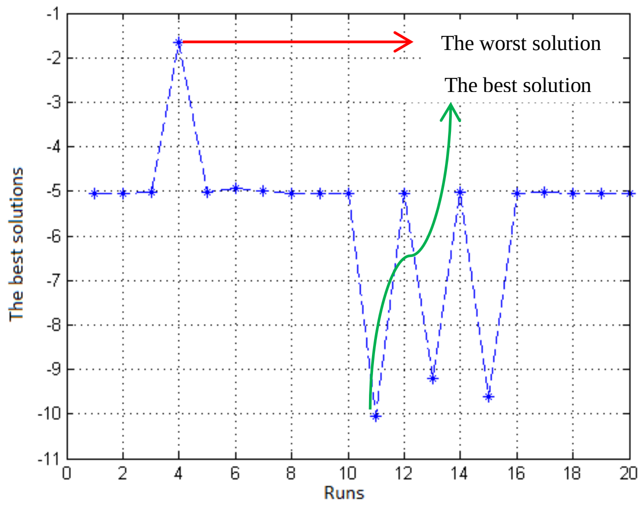
Figure 5.8: Performance of The PC Algorithm on Functions $F_{17} - F_{19}$.



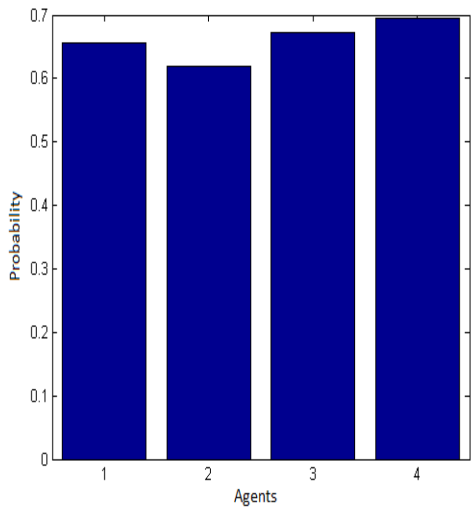
F_{20} (Hartman 6)



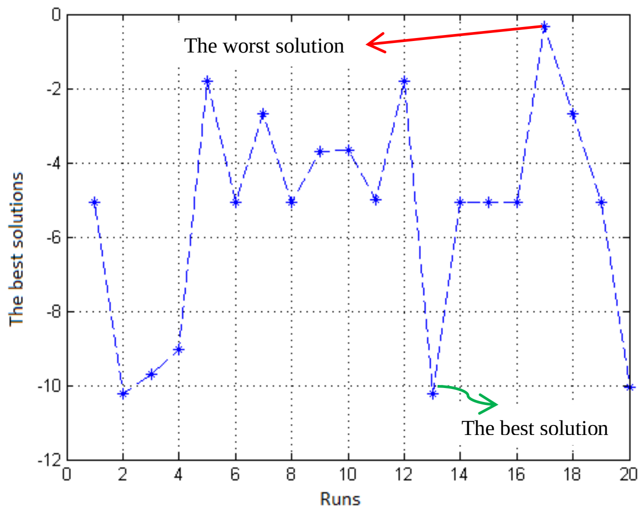
Favorable strategy of 6 agents at run 12



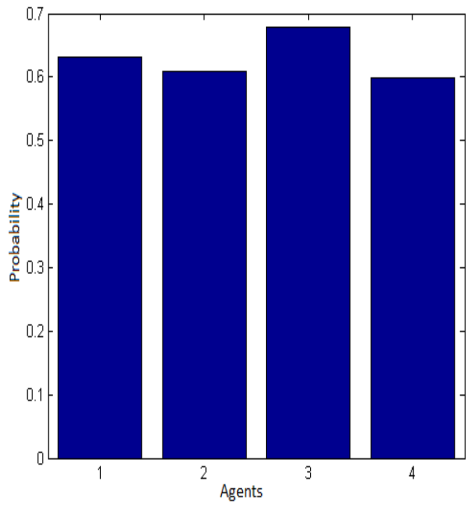
F_{21} (Shekel 5)



Favorable strategy of 4 agents at run 1



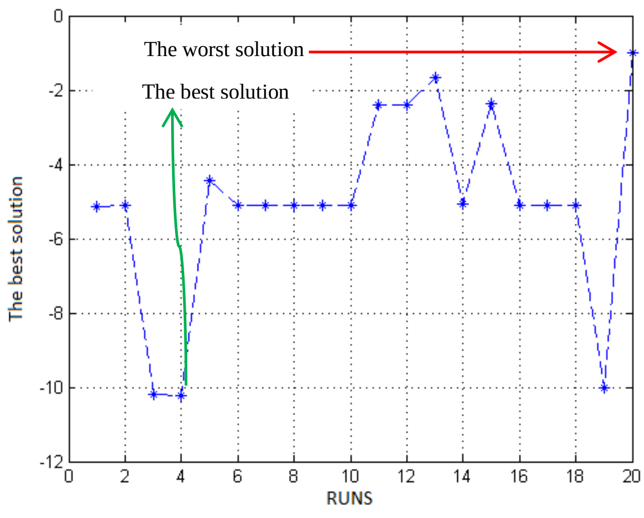
F_{22} (Shekel 7)



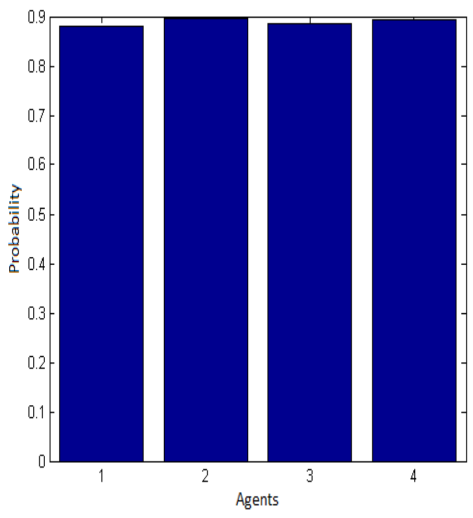
Favorable strategy of 4 agents at run 13

(a) Shows the best result over 20 runs (b) Shows favourable strategy of N agents

Figure 5.9: Performance of The PC Algorithm on Functions $F_{20} - F_{22}$.



F_{23} (Shekel 10)



Favorable Strategy of 4 agents at run 4

(a) Shows the best result over 20 runs (b) Shows favourable strategy of N agents

Figure 5.10: Performance of The PC Algorithm on Functions F_{23}.

In the following, the summary of all functions is demonstrated in table 5.1. All outcomes have been averaged over 20 runs. We found the mean best functions values over 20 times, and standard deviation. Moreover, we took the minimum best solutions for all runs and the maximum best solutions over 20 runs. It can be said from table 5.1 that the majority varies between functions $F_1 - F_7$, $F_8 - F_{13}$, and $F_{14} - F_{23}$ is that functions $F_{14} - F_{23}$ are convergence rate faster than other functions, because these functions have low dimensions and small local minima.

Table 5.1: PC Results of 23 Benchmark Problems

Functions	N	S	Number of Generations	Best solution	Worst solution	Mean	STD DEV	Time	Optimum
F1	30	[-100,100]	2000	1.0797e-07	1.975e-05	4.0497e-06	5.6421e-06	171.194 s	0
F2	30	[-10,10]	2000	1.2452e-06	0.0006045	6.0565e-05	0.00014377	256.867 s	0
F3	30	[-100,100]	2000	4.6356e-08	8.0209e-05	6.4234e-06	1.7668e-05	275.03 s	0
F4	30	[-100,100]	2000	6.8104e-07	0.00012707	1.4142e-05	2.8366e-05	365.561 s	0
F5	30	[-30,30]	2000	28.7648	29.0015	28.9699	0.0695	2061.81 s	0
F6	30	[-100,100]	2000	0	21	4.7	5.526	180.40 s	0
F7	30	[-1.28,1.28]	2000	1.2743e-05	0.0021258	0.00066206	0.0005144	2543.27 s	0
F8	30	[-500,500]	1000	-12435.8614	-1520.0505	-5673.9477	3015.9275	1690.48 s	-12569.5
F9	30	[-5.12,5.12]	1000	1.3225e-07	0.00025839	2.152e-05	6.0401e-05	145.41s	0
F10	30	[-32,32]	1000	3.7471e-07	0.00015753	1.2967e-05	3.4556e-05	312.60 s	0
F11	30	[-600,600]	1000	8.0909e-08	6.0028e-05	5.2176e-06	1.3048e-05	178.087 s	0
F12	30	[-50,50]	1000	0.1054	0.15941	0.14468	0.018089	3535.41 s	0
F13	30	[-50,50]	1000	0.28619	0.3	0.29628	0.0039995	3307.92 s	0
F14	2	[-65.536,65.536]	1000	1.0375	16.6154	9.3546	4.6737	182.467 s	1
F15	4	[-5,5]	1500	0.00049479	0.14844	0.062034	0.06317	307.88 s	0.0003075
F16	2	[-5,5]	100	-1.0316	-0.99236	-1.0103	0.01411	32.62 s	-1.0316285
F17	2	[-5,10]x[0,15]	500	0.3981	19.5348	1.3624	4.2773	88.4769 s	0.398
F18	2	[-2,2]	1000	3.0001	599.06	119.09	179.64	125.03 s	3
F19	4	[0,1]	500	-3.4545	-0.7997	-2.395	0.81776	69.99 s	-3.86
F20	6	[0,1]	1000	-3.3196	-0.65012	-2.847	0.8126	204.63 s	-3.32
F21	4	[0,10]	2000	-10.04	-1.6524	-5.5518	1.9146	100.03 s	-10
F22	4	[0,10]	2000	-10.24	-0.337	-5.322	3.017	73.81 s	-10
F23	4	[0,10]	2000	-10.215	-0.996	-5.043	2.590	74.023 s	-10

5.2 Results of EL Farol Bar Problem

The results of this problem being repeated 20 runs as shown in figure 5.11. We found the attendance at the bar in each of the 1000 weeks. The mean was not stable as in figure 5.11. It is noteworthy that, the average was approximately among 54 and 53.

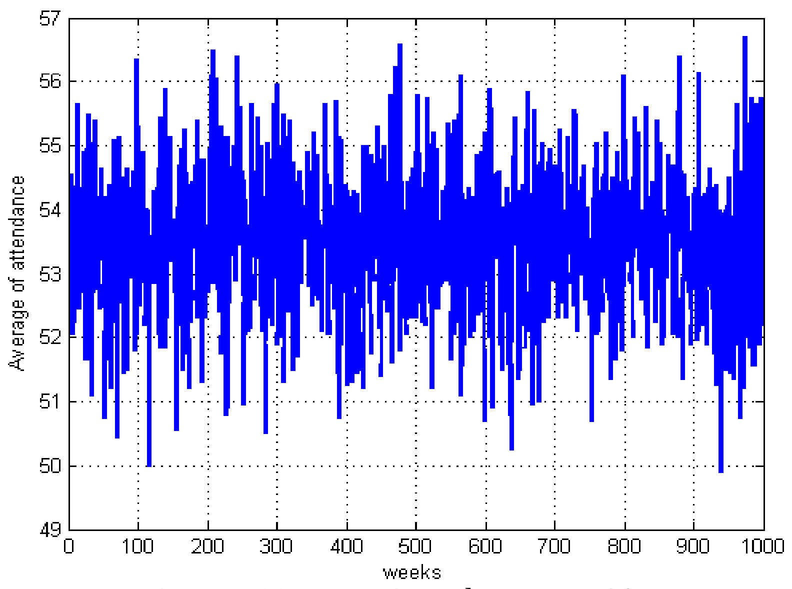


Figure 5.11: Mean Attendance over 20 Runs

Figure 5.12 illustrates the standard deviation of attendance at the bar over 20 runs. There were fluctuations of standard deviation for all weeks between 4.5 and 5.

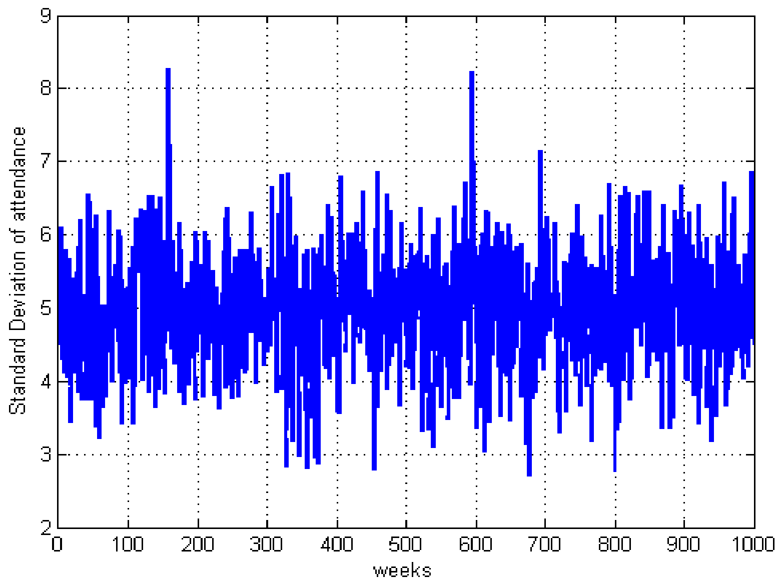


Figure 5.12: Standard Deviation of Attendance over 20 Runs.

Figure 5.13 is shown the last attendance at the bar. The number of attendance is fluctuated to among 55 and 67 for each week. Attendance in 5 weeks is 64. This means, the bar is crowded, whereas attendance in the first week is about 46 attendances. This means, agent attends the bar. The results of each trial were similar with that illustrated in figure 5.13.

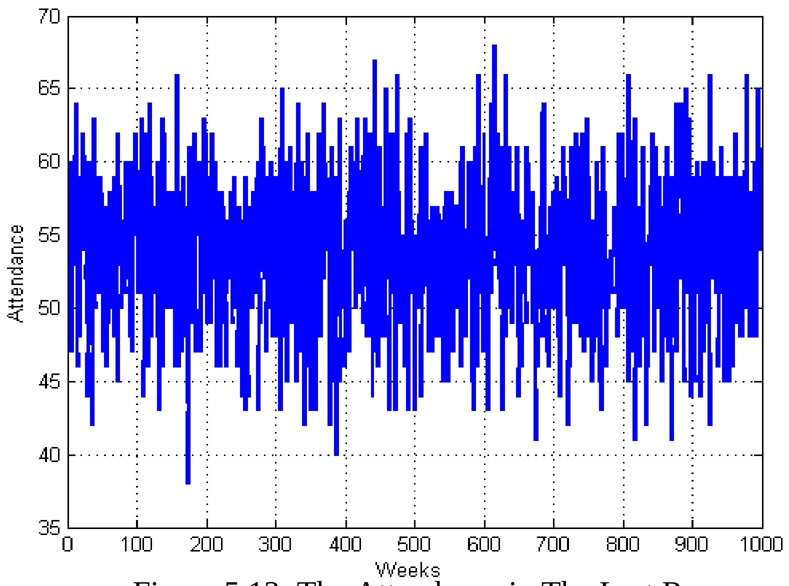
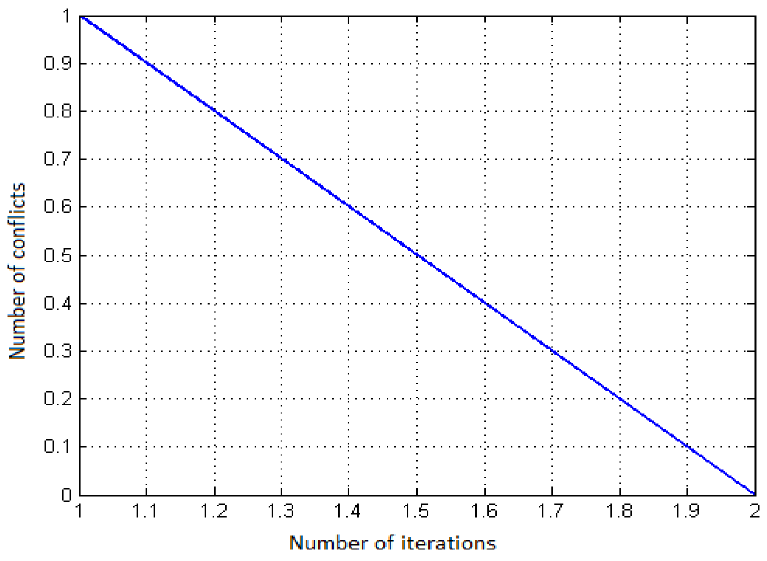


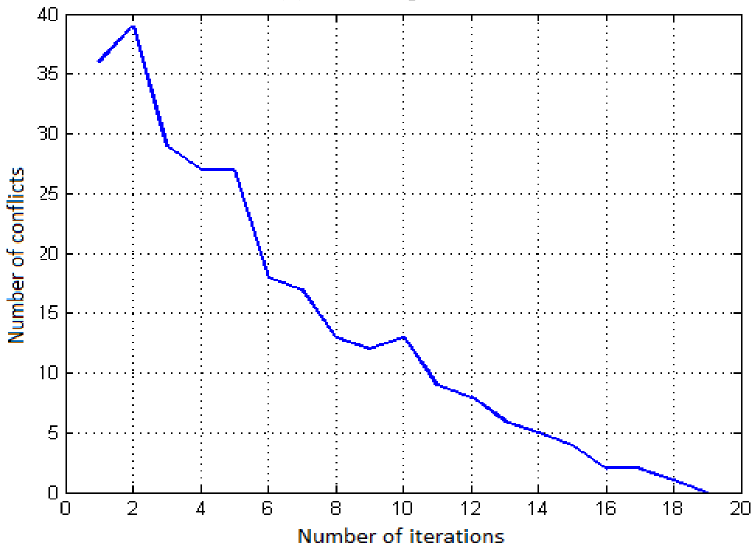
Figure 5.13: The Attendance in The Last Run.

5.3 Results of N-Queens Problem

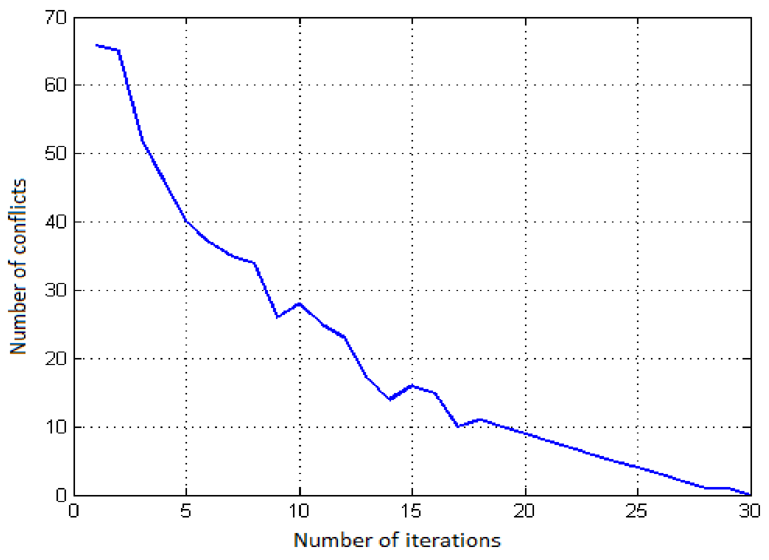
We implemented the PC algorithm to solve the N-queens problems at different sizes ($N = 8,100,150,180$). Figure 5.14 illustrates the convergence for various sizes of problem. The numbers of conflicts drop sharply from 1 to 0 as in figure 5.14 (a). However, the number of conflicts in figure 5.14 (b, c, d) decreased gradually to reach zero. This means, the small size of problem took much less iterations to convergences than larger sizes.



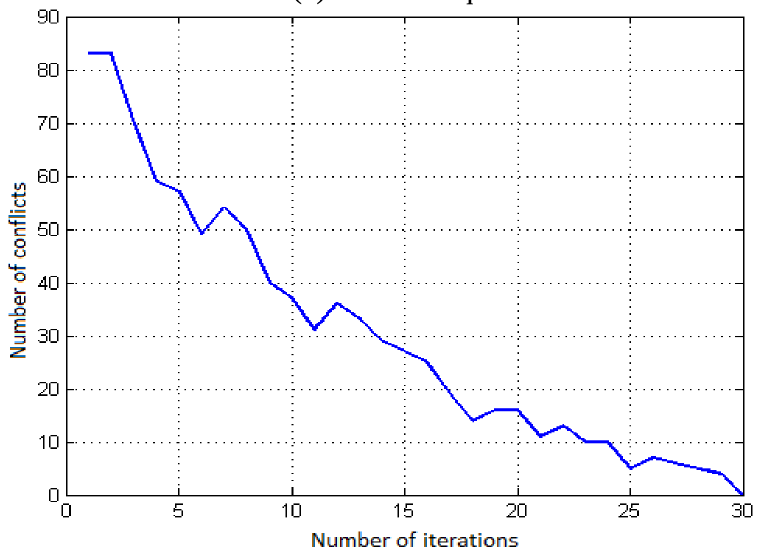
(a) N = 8 queens



(b) N = 100 queens



(c) N = 150 queens



(d) N = 180 queens

Figure 5.14: (a-d) The Convergence for PC Algorithm Runs on Different Sizes of Queens.

Table 5.2 shows the result of applying the PC algorithm to find the best solution for N-queens, with the time that is needed to achieve each solution. For example, 8 queens are distributed on chessboard as :(q_1 at column 4 , q_1 at column 8, q_1 at column 1, q_1 at column 5, q_1 at column 7, q_1 at column 2, q_1 at column 6 , q_1 at column 3).

Table 5.2: The Results of N-Queens Problem and Time of Reaching The optimum Solutions

Number of queens	Number of Generations	Time	Solutions
8	2	0.408623 s	[4,8,1,5,7,2,6,3]
100	19	2168.79359s	[27,90,17,62,47,2,41,71,24,3,80,4,37,66,96,87,29,69,28,65,67,1,54,57,84,94,40,44,89,95,11,73,31,34,19,38,98,100,45,76,8,52,46,14,53,36,77,42,81,78,56,5,10,79,18,43,13,93,97,55,83,51,15,63,85,70,33,91,6,99,68,21,64,48,82,20,25,23,7,39,92,59,12,22,49,26,88,60,32,75,72,74,61,35,50,30,58,16,9]
150	30	6067.604771s	[34,140,61,80,49,147,137,148,40,21,23,48,127,24,31,81,68,55,3,97,133,67,120,135,44,87,96,100,33,89,2,25,14,116,130,9,5,136,11,142,129,94,45,76,107,63,113,53,56,79,126,4,43,122,149,111,6,57,20,26,46,42,144,131,118,17,146,65,77,10,134,58,38,98,64,110,124,93,70,47,27,109,8,114,108,95,105,138,117,132,60,22,19,1,71,145,13,90,86,18,37,39,128,74,125,16,30,102,36,92,15,12,139,73,75,78,112,62,115,28,121,35,52,7,99,32,91,50,85,119,82,66,69,54,83,59,141,84,29,104,143,41,123,88,51,101,72,106]
180	30	30240.05265s	[34,176,103,170,147,10,68,142,73,97,122,41,35,172,155,65,109,61,168,171,123,137,62,134,100,108,37,46,133,11,5,1,95,28,70,78,159,57,47,124,148,143,39,24,135,4,48,52,140,71,119,141,113,150,116,6,74,144,158,76,83,88,14,161,110,23,55,96,69,44,20,84,127,120,85,111,25,131,51,12,87,139,153,178,160,152,128,66,18,13,19,27,90,7,22,145,164,180,17,45,54,101,59,129,174,177,16,126,72,166,36,3,33,82,99,26,175,21,156,179,107,43,165,162,115,9,89,104,121,163,138,86,75,132,102,151,92,15,2,117,114,40,130,63,56,98,80,49,42,93,154,118,112,58,94,64,146,53,105,125,91,30,79,149,136,8,32,38,29,60,173,50,167,31,81,157,77,169,67]

5.1 Results of Repeated Game Problems

We carried out the PC algorithm on a different repeated games such as (Prisoner's dilemma, Stag hunt, Battle of sexes game, Choose sides), with the number of runs equal to 500 and the number of agents 10, starting from random initial values. We had two results for each problem. The first graph illustrates the mean of payoff at each ran. The second graph shows the decimal representation of values at each run. We discussed result each problem as follows:

- **Results of Prisoner's Dilemma game**

We can see in figure 5.15 that the mean payoffs converged gradually to a value near 1 at 150 iterations. It is clear that the Prisoner's Dilemma simulations stabilized even after 150 runs. Thus, the optimal strategy is for all players to defect (D, D), yielding payoffs of (1, 1). Figure 5.16 shows the decimal representation of the best value. We can see that after around 20 runs this value decreased to zero and stayed there.

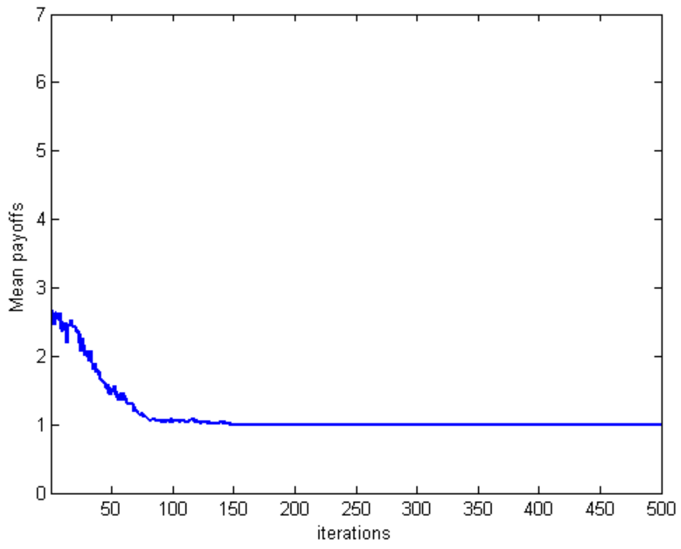


Figure 5.15: The Mean of Payoffs over 500 Iterations (Prisoner's Dilemma)

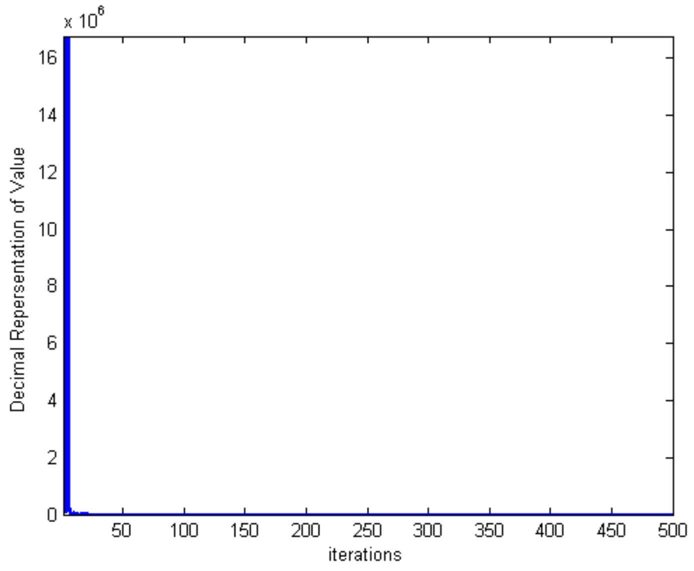


Figure 5.16: The Decimal Representation of The Best Value (Prisoner’s Dilemma)

- **Results of Stag Hunt game**

The average of payoffs remained stable to a value 1 for all iterations. The equilibrium point of stag hunt game is (Rabbit, Rabbit) which is easy to achieve because of risk dominant the highest possible payoff as shown in figure 5.17. The best value declined from 1×10^{24} to zero at iteration 10 as shown in the figure below 5.18. Thus, the best strategy of this game was always defection “hunting rabbits”(R, R).

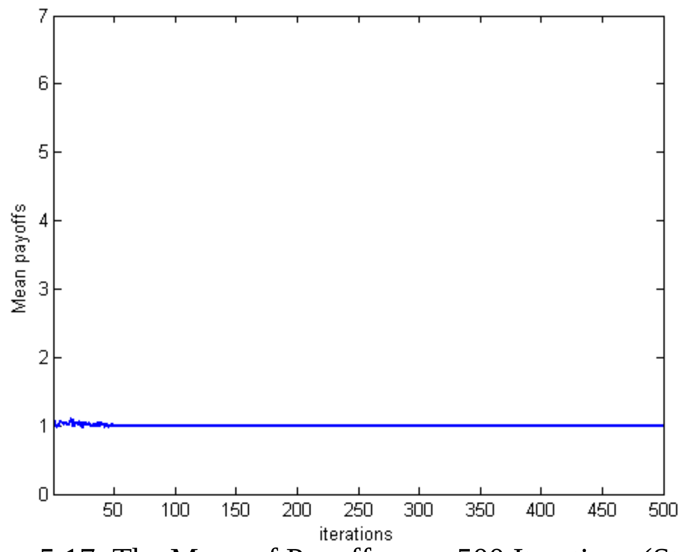


Figure 5.17: The Mean of Payoffs over 500 Iterations (Stag Hunt)

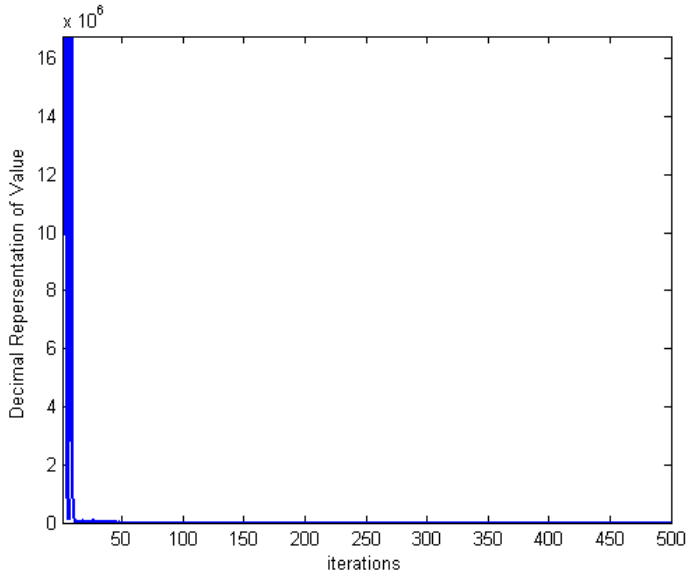
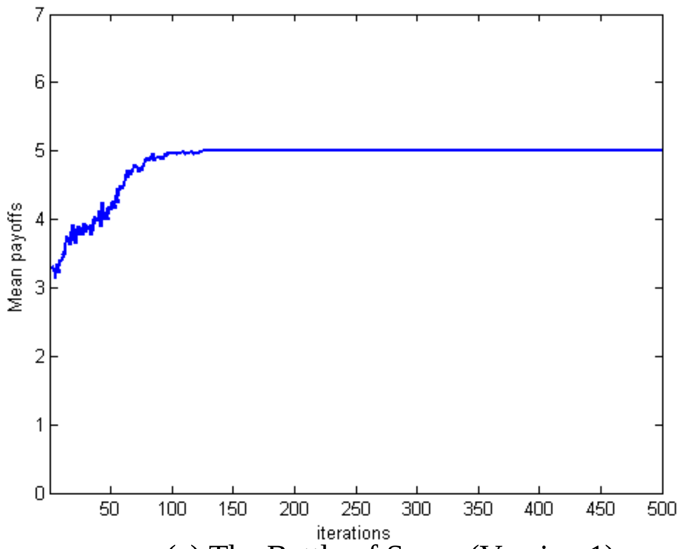


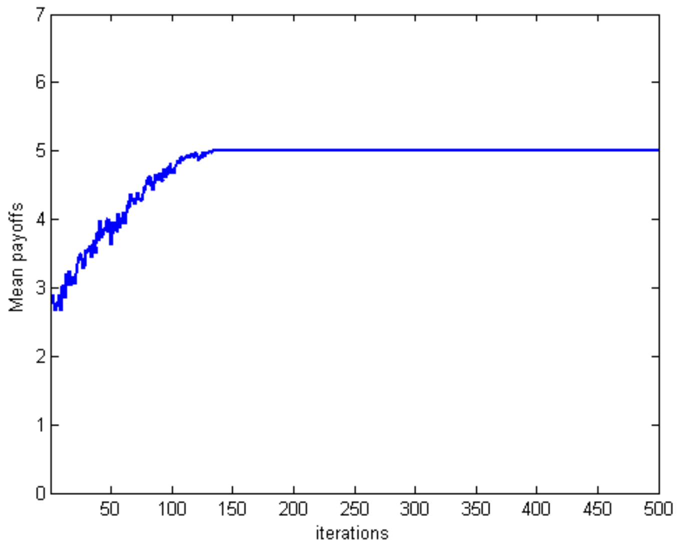
Figure 5.18: The Decimal Representation of The Best Value (Stag Hunt)

- **Results of Battle of Sexes Game**

This game has two versions. The results of both versions were inherently similar. The average of payoffs converged at around 150 runs in version 1 to a value 5 and about 105 runs in version 2 to a value 5 as shown in figure 5.19. The graph shows the most outcomes to have been two equilibrium points of this game, one at (football, football) and one at (opera, opera). In figure 5.20, the best value of version 1 decreased at around 15 iterations to zero. Whereas, the best value of version 2 dropped approximately 10 times to zero.

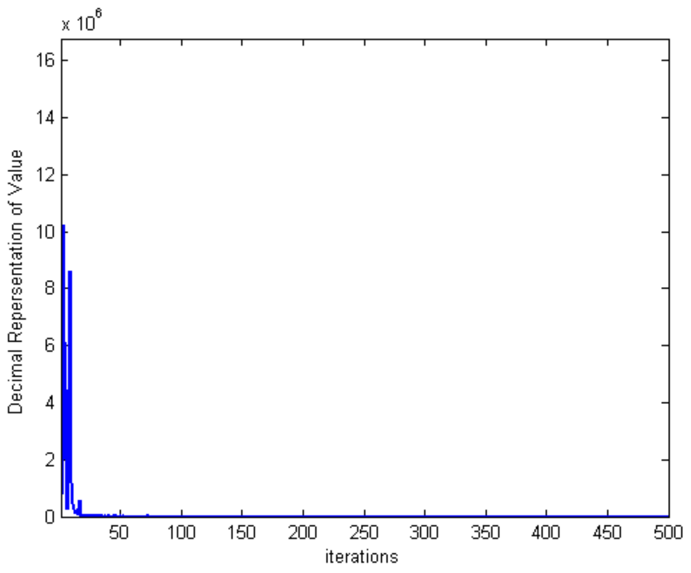


(a) The Battle of Sexes (Version 1)

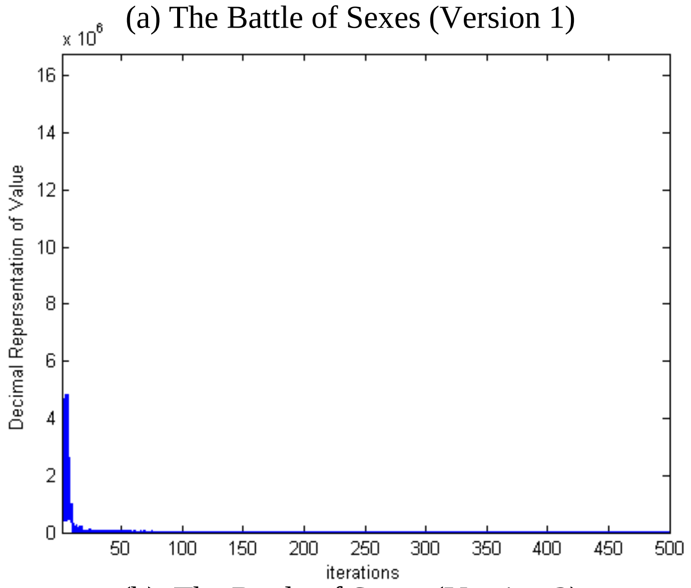


(b) The Battle of Sexes (Version 2)

Figure 5.19: The Mean of Payoffs over 500 Iterations (Battle of Sexes Game)



(a) The Battle of Sexes (Version 1)



(b) The Battle of Sexes (Version 2)

Figure 5.20: The Decimal Representation of The Best Value (The Battle of Sexes)

- **Results of Choosing Sides Game**

In this game requires the two players to choose the same action both right or both left. In figure 5.21, we can see that the convergence of the average payoffs was close to one at 100 runs and remained the same value after this iteration, which reflects the payoffs at the two equilibrium (Right, Right) and (Left, Left). The decimal representation of value converged to zero at 10 iterations as in figure 5.22.

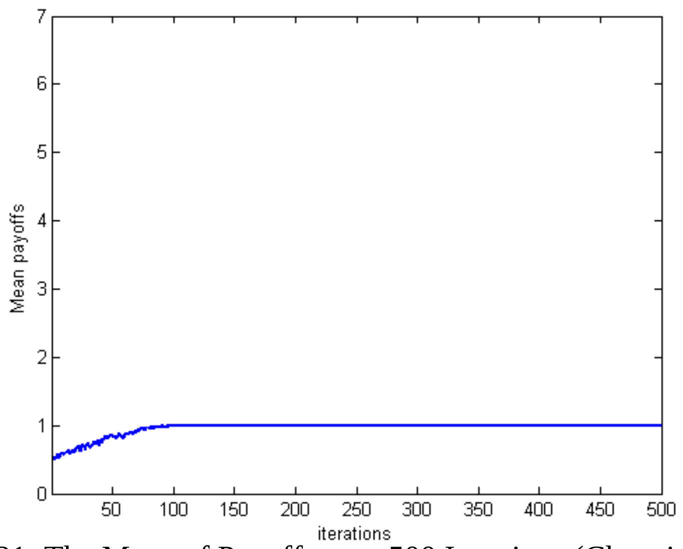


Figure 5.21: The Mean of Payoffs over 500 Iterations (Choosing Sides)

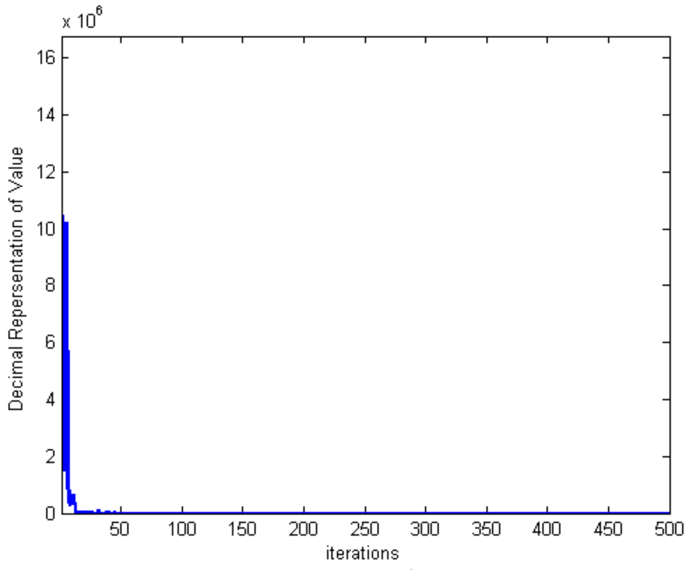


Figure 5.22: The Decimal Representation of The Best Value (choosing Sides)

To sum up, we compared the results of four games. We found that the best result was the stage hunt game because the convergence of this game was faster than other. Whereas, the convergence of others games was inherently similar.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

In this thesis, we have implemented several problems using probability collectives algorithm. The PC is a general framework of agent coordination and distributed optimisation, which is a type of heuristic algorithm. This algorithm concentrates on adapting the distributions the strategy set of each agent to improve its performance. Each agent makes options using the determined utility until the algorithm reaches the convergence. The performance of probability collective is tested using four problems such as 23 benchmark problems, El Farol bar problem, N-queens problem and repeated games problems. The results show that the PC algorithm was successful and was sufficiently robust in solving these problems.

6.2 Future Work

In the future task, we will implement PC algorithm to solve the multi-objective optimisation for all problems the proposed in this thesis. We will also hybridise PC algorithm with different technics such as simulated annealing to estimate expected utility and escape from a local minimum. Furthermore, we will develop parameters of PC algorithm such as the number of strategies M_i and number of iterations.

REFERENCES

[1] Schut, M. C. (2010). On model design for simulation of collective intelligence. *Information Sciences*, *180*(1), 132-155.

[2] Zhao, W., & Wang, N. (2013, June). Probability Collectives using Response Surface estimation. In Communications and Information Technology (ICCIT), 2013 Third International Conference on (pp. 1-5). IEEE.

[3] Kulkarni, A. J., & Tai, K. (2008, October). Probability collectives for decentralized, distributed optimization: a collective intelligence approach. In Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on (pp. 1271-1275). IEEE.

[4] Kulkarni, A. J., & Tai, K. (2010). Probability collectives: a multi-agent approach for solving combinatorial optimization problems. *Applied Soft Computing*, *10*(3), 759-771.

[5] Bieniawski, S., Wolpert, D. H., & Kroo, I. (2004, September). Discrete, continuous, and constrained optimization using collectives. In Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York.

[6] Huang, C. F. (2012). Combinatorial optimization in biology using probability collectives multi-agent systems. *Expert Systems with Applications*, *39*(2), 1763-1771.

[7] Kulkarni, A. J., & Tai, K. (2010, July). Probability collectives: a distributed optimization approach for constrained problems. In *IEEE Congress on Evolutionary Computation* (pp. 1-8). IEEE.

[8] Kulkarni, A. J., Tai, K., & Abraham, A. (2015). *Probability Collectives: A Distributed Multi-agent System Approach for Optimization* (Vol. 86). Springer.

[9] Deb, K., & Tiwari, S. (2008). Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, *185*(3), 1062-1087.

[10] “Multi-agent system”, WIKIPEDIA, [Online].Available:“ http:// en.wikipedia.org/wiki/Multi-agent_system”.

[11] Tweedale, & Jeffrey, 3 (2007). Innovations in multi-agent systems. *Journal of Network and Computer Applications* 30.3, 1089-1115.

[12] Faltings, B., & Nguyen, Q. H. (2005, July). Multi-agent Coordination using Local Search. In *IJCAI* (pp. 953-958).

[13] Xu, Z., Unveren, A., & Acan, A. (2016). Probability collectives hybridised with differential evolution for global optimisation. *International Journal of Bio-Inspired Computation*, *8*(3), 133-153.

[14] Yang, B., & Wu, R. (2016). A modified probability collectives optimization algorithm based on trust region method and a new temperature annealing schedule. *Soft Computing*, *20*(4), 1581-1600.

[15] Lam, A. Y., Li, V. O., & James, J. Q. (2012). Real-coded chemical reaction optimization. *IEEE Transactions on Evolutionary Computation*, *16*(3), 339-353.

[16] Rand, W., & Stonedahl, F. (2007). The El Farol Bar Problem and Computational Effort: Why People Fail to Use Bars Efficiently. *Northwestern University, Evanston, IL*.

[17] Vasirani, M., & Ossowski, S. (2007, October). Collective-based multiagent coordination: a case study. In *International Workshop on Engineering Societies in the Agents World* (pp. 240-253). Springer Berlin Heidelberg.

[18] Cull, P., & Pandey, R. (1994). Isomorphism and the n-queens problem. *ACM SIGCSE Bulletin*, *26*(3), 29-36.

[19] "Nash Equilibrium", WIKIPEDIA, [Online]. Available: "https://en.wikipedia.org/wiki/Nash_equilibrium".

[20] Mittal, S., & Deb, K. (2009). Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives. *IEEE Transactions on Evolutionary Computation*, *13*(3), 554-565.

[21] “Prisoner's Dilemma”, WIKIPEDIA, [Online]. Available: “https://en.wikipedia.org/wiki/Prisoner%27s_dilemma”

[22] Kimbrough, S. O. (2005, May). Foraging for trust: Exploring rationality and the stag hunt game. In International Conference on Trust Management (pp. 1-16). Springer Berlin Heidelberg.

[23] Browning, L., & Colman, A. M. (2004). Evolution of coordinated alternating reciprocity in repeated dyadic games. *Journal of Theoretical Biology*, *229*(4), 549-557.

[24] “Coordination Game”, WIKIPEDIA, [Online]. Available: “https://en.wikipedia.org/wiki/Coordination_game”

[25] David Hales. (2012). Agent and Based Modelling in NetLogo: “http://davidhales.name/abm-netlogo/lab-el-farol/netlogo-abm-lists2-with.pptx.pdf”.

[26] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering, 186(2), 311-338.

[27] Ray, T., TAI, K., & SEOW, K. C. (2001). Multiobjective design optimization by an evolutionary algorithm. Engineering Optimization, 33(4), 399-424.