A Quantum Evolutionary Approach to Solving the Team Formation Problem in Social Networks

Master's Thesis presented by

Pedro Pablo Álvarez Lois

under the supervision of

Dr. Severino Fernández Galán

Máster Universitario en Inteligencia Artificial Avanzada

Escuela Técnica Superior de Ingeniería Informática

Universidad Nacional de Educación a Distancia

Final version: 4 September 2019

Abstract

Recent advances in information and communication technologies have led to the expansion of collaborative work. Complex problems in science, engineering, or business are being solved by teams of people working closely with one another. However, forming teams of experts is a computationally challenging problem that requires powerful solution techniques. A meta-heuristic algorithm that incorporates some of the principles of quantum computing into an evolutionary structure is presented. The resulting Quantum Evolutionary Algorithm (QEA) has the ability to produce an adequate balance between intensification and diversification during the search process. Numerical experiments have shown that the QEA is able to significantly improve the quality of the solutions for hard instances of the team formation problem, particularly when compared to a standard genetic algorithm. The successful performance of the algorithm requires careful parameter tuning, as well as a mechanism to effectively share information across the population of candidate solutions.

Para María, Rodrigo y Manuel

A la memoria de mis padres, Pedro y Rosa

Agradecimientos

Me gustaría agradecer sinceramente el trabajo de dirección llevado a cabo por el Profesor Severino Fernández Galán. Sus comentarios y sugerencias han sido detallados, precisos y realmente útiles. Sin duda, me han servido para materializar una idea inicial en un trabajo de investigación del que me siento verdaderamente orgulloso. Esto no sería posible sin su apoyo y dedicación. Muchas gracias, Severino.

A la Universidad Nacional a Distancia (UNED), por haberme dado la oportunidad de formar parte de esta Institución como alumno. Mi padre en su día también lo fue, precisamente al poco de fundarse la UNED. Eran tiempos complicados y nunca pudo terminar sus estudios. Ciertamente, ha habido momentos donde yo he estado a punto de desistir. Sin embargo, su recuerdo me ha dado ese extra de empuje necesario para continuar hasta el final y, con ello, poderle rendir un pequeño homenaje.

También querría hacer especial mención a mi familia. A mi esposa María, por su comprensión y generosidad al tener que lidiar con dos pequeñas y hermosas fieras mientras yo empleaba el tiempo “masterizando”. A mi hijo Manuel, que nació justo un mes antes de que yo comenzase el Máster y que hacía que me preguntase: ¿Por qué habré decidido meterme en esto ahora? Y, finalmente, a mi hijo Rodrigo que me preguntaba: Papá, ¿qué son esos cálculos tan raros que estás haciendo? y que me obligaba a darle una respuesta imaginativa, no siempre fácil.

"In the beginning there were only probabilities. The universe could only come into existence if someone observed it. It does not matter that the observers turned up several billion years later. The universe exists because we are aware of it."

Martin Rees
Astronomer Royal

1st Law of Computing: You cannot solve uncomputable or NP- hard problems efficiently unless you have a physical infinity or an efficient oracle.

2nd Law of Computing: There are no physical infinities or efficient oracles.

3rd Law of Computing: Nature is physical and does not solve uncomputable or NP-hard problems efficiently.

Corollary: Nature necessarily solves uncomputable or NP-hard problems only approximately.

Göran Wending (2019)

Table of Contents

LIST OF FIGURES **2**

LIST OF TABLES **3**

CHAPTER 1 INTRODUCTION **4**

1.1 MOTIVATION AND OBJECTIVE 4
1.2 METHODOLOGY 6
1.3 CONTRIBUTION OF THIS WORK 10
1.4 ORGANISATION OF THE THESIS 10

CHAPTER 2 TEAM FORMATION PROBLEM IN SOCIAL NETWORKS **11**

2.1 INTUITIVE DESCRIPTION 11
2.2 FORMAL DEFINITION 14

CHAPTER 3 REVIEW OF THE LITERATURE **17**

3.1 REVIEW OF APPROXIMATION METHODS 18
3.2 REVIEW OF METAHEURISTIC METHODS 20
3.2.1 Evolutionary Algorithms *20*
3.2.2 Other Metaheuristic Methods *22*

CHAPTER 4 DESIGN OF A QUANTUM EVOLUTIONARY ALGORITHM FOR THE TEAM FORMATION PROBLEM IN SOCIAL NETWORKS **24**

4.1 EVOLUTIONARY ALGORITHM 24
4.2 THE QUANTUM EVOLUTIONARY ALGORITHM 26
4.2.1 Quantum-bit Representation *27*
4.2.2 Main Components of QEAs *29*
4.3 THE QEA FOR THE TFPSN 33
4.3.1 Initialisation *33*
4.3.2 Main Loop *35*
4.3.3 Termination *44*
4.4 FORMULATION OF THE TFPSN IN THE CONTEXT OF THE QEA 45
4.4.1 Representation of the Problem *45*
4.4.2 Definition of the Fitness Function *46*

CHAPTER 5 EVALUATION **48**

5.1 EXPERIMENTAL SETUP 49
5.2 EVALUATION OF A GENETIC ALGORITHM 52
5.2.1 Configuration of the Parameter Values *53*
5.2.2 Results *54*
5.3 EVALUATION OF THE QEA 62
5.3.1 Baseline Specification *63*
5.3.2 Time-varying Q-Gate Rotation Angles *67*
5.3.3 Analysis of Convergence *71*
5.3.4 Alternative Attraction Strategies *73*

CHAPTER 6 CONCLUDING REMARKS AND FUTURE RESEARCH **76**

BIBLIOGRAPHY **79**

List of Figures

Figure 2.1 A Social network of experts and their collaboration costs 12
Figure 2.2 Two feasible teams for the task T= {s_1, s_2, s_3, s_4, s_5}.. 13
Figure 3.1 Classification of optimisation algorithms.. 17
Figure 4.1 The evolutionary cycle 25
Figure 4.2 Different types of representations for individuals in GAs 29
Figure 4.3 Pseudo-code for the QEA... 30
Figure 4.4 Block diagram for the QEA.. 31
Figure 4.5 Block diagram for the canonical GA 32
Figure 4.6 Polar plot for a Q-bit individual.. 37
Figure 4.7 Polar plot of vector rotations 38
Figure 4.8 Polar plot of the rotation gate for a Q-bit individual... 39
Figure 4.9 Illustration of a local migration strategy... 43
Figure 4.10 Illustration of a global migration strategy 44
Figure 4.11 Overall structure of QEAs as described in Han and Kim (2002).......................... 45
Figure 5.1 Randomly generated pool of 10 experts 50
Figure 5.2 Sparse graph with 10 experts and a cost threshold of 0.6...................................... 50
Figure 5.3 Graph of a pool consisting of 30 experts... 51
Figure 5.4 Problem 1: Progress Curve 55
Figure 5.5 Problem 1: Hamming distance 56
Figure 5.6 Problem 5: Progress curve 57
Figure 5.7 Problem 5: Hamming distance 57
Figure 5.8 Problem 5: Best cost for different Tournament sizes 58
Figure 5.9 Problem 5: Hamming distance vs. Tournament size 59
Figure 5.10 Problem 5: Best cost for different Crossover rates 60
Figure 5.11 Problem 5: Hamming distance vs. Crossover rate 61
Figure 5.12 Problem 5: Best cost for different Mutation rate 61
Figure 5.13 Problem 5: Hamming distance vs. Mutation rate.. 62
Figure 5.14 Problem 4: Convergence QEA for different rotation parameters 65
Figure 5.15 Problem 4: Minimum cost for varying rotation parameters $\theta 1$ and $\theta 2$............... 66
Figure 5.16 Problem 4: Contour plot for varying rotation parameters $\theta 1$ and $\theta 2$.................. 66
Figure 5.17 Problem 4: Progress Curve QEA with time-varying $\theta 1$ and fixed $\theta 2$ 69
Figure 5.18 Problem 4: Convergence QEA with time-varying $\theta 1$ and fixed $\theta 2$ 70
Figure 5.19 Problem 4: MSE Curve for QEA with time-varying $\theta 1$ and fixed $\theta 2$ 72
Figure 5.20 Problem 4: Quantiles for QEA with time-varying $\theta 1$ and fixed $\theta 2$ 73
Figure 5.21 Best collaboration cost: comparison QEA vs. GA 75

List of Tables

Table 2-1 Notation for the TFPSN 14
Table 5-1 Configurations of the TFPSN considered in the analysis 52
Table 5-2 Parameter values for the baseline genetic algorithm 52
Table 5-3 Results for the baseline genetic algorithm 54
Table 5-4 Look-up table for baseline QEA with rotation parameters $\theta 1$ and $\theta 2$........ 63
Table 5-5 Problem 4: Minimum cost with rotation parameters $\theta 1 = \theta 2$........ 64
Table 5-6 Problem 4: minimum cost varying rotation parameters $\theta 1$ and $\theta 2$ 67
Table 5-7 Problem 4: minimum cost with varying rotation as in Wang et al. (2014) 68
Table 5-8 Problem 4: minimum cost with varying rotation as in Lahoz-Betra (2016)........ 71
Table 5-9 Results for QEA with global attractor and adaptive parameters $\theta 1$and $\theta 2$ 74

Chapter 1 Introduction

1.1 Motivation and Objective

Collaborative problem solving has been the subject of artificial intelligence research for almost half a century as reflected in the pioneering work of Newell and Simon (1972). Gradually, the focus of artificial intelligence research changed from trying to replicate human intelligence to augment human capabilities, both at individual as well as at the societal level. With advancements in computing and communication technologies, collectives of intelligent agents have been applied to problems in varied domains like optimisation tasks, the Web and social systems. As mentioned in Singh and Gupta (2009) and discussed furtherly in Easley and Kleinberg (2010), investigations in human computation, social computing and complex networks have also contributed to new results, tools and technologies for social and human problem solving.

There are many dimensions in which the collective use of resources yields better solutions to a specific problem or task. In this sense, it must be noted that intelligence does not arise only in individual brains; it also arises in groups of individuals. This is a fundamental concept and is known as *collective intelligence*: groups of individuals acting collectively in ways that seem intelligent. The result that emerges from the synergistic aggregation of individuals' potentials is larger than any individual's potential or a mere sum of the individuals' potentials.

The so-called wisdom of crowds, discussed at length in Surowiecki (2004), is a simple idea that has profound implications: a large collection of people is smarter than an elite few at solving problems, fostering innovation, coming to wise decisions, and even deciding how our social and economic activities should be organised. In this regard, Theiner et al. (2009) and Wooley et al. (2015) show that groups of people can manifest cognitive capacities beyond the simple aggregation of their individual members. Awal and Bharadwaj (2018a) give an overview of the principles of collective intelligence and the concept of wisdom of crowds and highlight, for instance, how to maximize the potential of big data analytics in this context.

Complex problems in science, engineering, or business are being solved by teams of people working closely with one another, each with the help of their network. As noted by Malone and Bernstein (2015), growing amount of scientific research is done, for instance, in an open collaborative fashion in projects sometimes referred to as "citizen science". Collaborative software design and prediction tasks are also examples of teams of people sharing their skills to perform a task or solve a problem.[1]

Effective organisations must thus have a structure that fosters the efficient formation of teams to accomplish difficult tasks. In collaborative work, groups of agents must co-ordinate effectively to solve problems, allocate tasks across a distributed organisation and achieve collective goals. Team formation is essential in this regard. The success of a team of agents working collaboratively to solve a given task or problem will depend to a large extent on the amount of the so-called group collective intelligence. As noted in Wi et al. (2009), this is made of essentially two key components: one is knowledge competence, that is, the skills of each of the members of the team which depends on her inherent intelligence as well as knowledge gained from interacting with other members of her social network; the other component is collaboration competence which refers to how well those agents interact with each other.

Two important issues associated with team members influence the success of a project: the expertise of team members and the quality of collaboration and communication among these people as a team. Regarding the first issue, a trend in this line of work is to model the problem as finding the optimal set of experts for the demanded functional requirements considering multiple aspects such as technical skills, cognitive characteristics, and personal motivation. Klug and Bagrow (2016), for instance, analyse how the team size, the way work is distributed among team members and the differences and similarities in the experiences and backgrounds of those team members affect the overall performance. Such studies emphasize only on properties of individual team members that are independent of the final team configuration. This type of problem has been applied, for instance, to team project selection. With their high potential, high motivation, great problem-solving ability and flexibility, project teams are important work structures for the business and scientific life. The success of these teams is highly dependent upon the people involved in the project team. This makes the project team selection an important factor for project success.

Because of the increasing use of the Internet, online *social networks of experts* have become popular. Increasingly, businesses look for subject matter professionals to successfully complete tasks. Although these approaches largely ignore the role of social network information, such information should not be disregarded. The problem of team formation

[1] An overview of this field is given in Franzoni and Sauermann (2014). Relevant applications include the study of climate change as described in Duhaime et al (2015) and the Polymath Project presented in Gowers and Nielsen (2009). Michelucci and Dickinson (2016) review applications in human-computer interaction. Nickerson (2015) illustrates applications in the field of collective design, where creative work is performed by thousands of people who interact through collective design systems. Kyriakou et al (2014) analyse open collaboration in hardware design.

should be extended to one in which the objective is to achieve an optimal allocation of agents or experts to specific tasks considering the interaction amongst team members.

The social graph structure of the individuals needs to be modelled explicitly. In this sense, Chhabra et al. (2012) consider the problem where experts on different task types may have a synergistic effect on each other's performance when they are allocated to the same project by fruitfully sharing information that could be valuable for different task types. The authors demonstrate that taking social network structure into account can have significant benefits in terms of the overall optimality of task allocation. Dykhuis et al. (2013) explored the performance of a simple, agent-based team-formation strategy in a variety of graph topologies and task structures. Even though they do not achieve optimal performance, small-world graph topologies, which reflect characteristics of real social networks, are efficient structures for team formation. Because they have a limited number of local connections, convergence is very fast, but because of the small-world structure, agents have quick access to a majority of the other agents, leading to efficient use of these few connections in finding effective solutions.

The goal of the present research work is to develop a new method to form the best team of experts in a social network to accomplish a task in the most efficient manner. This problem is known in the literature as the Team Formation Problem in Social Networks (TFPSN). There are two main reasons that motivate the interest in this problem. First, it is very relevant as demonstrated by recent advances in information and communication technologies that have led to the expansion of collaborative work in many disciplines; second, it is computationally difficult and hence requires powerful solution techniques.

1.2 Methodology

Team formation is a well-known and widely studied combinatorial optimisation problem. The main reference in the field is the work of Lappas et al. (2009). These authors are the first to consider the problem of team formation within social networks. They defined the problem of team formation as follows: Finding a team of experts that not only cover the required skills of the task, but also the members can work effectively with each other. The social network is modelled as a graph where each node represents an expert, with one or more skills, and each edge is weighted by the collaboration cost between the two corresponding experts. These authors also proved that the problem of team formation is a complex one, and more specifically, an NP-hard. This means that no polynomial time algorithm exists that finds the optimal solution for all specific instances of this problem.

1.2.1 General Approach

The TFPSN is a challenging problem that has been addressed in the literature from different angles, particularly using near-optimal methods. These algorithms produce high-quality solutions in a reasonable time for real-world applications, but there is no guarantee of discovering a global optimal solution. Within this approach, two categories of algorithms can be distinguished: approximation and metaheuristic algorithms.

Approximation algorithms provide provable solution quality and provable run-time limits. Hence, the design and analysis of these algorithms involves a mathematical proof certifying the quality of the returned solutions in the worst case. This distinguishes them from heuristics, which find reasonably good solutions on some inputs, but provide no clear indication at the outset on when they may succeed or fail. Also, many metaheuristics implement some form of stochastic optimization, so that the solution found is dependent on the set of random variables generated. In contrast, approximation methods tend to be non-deterministic, although some algorithms incorporate random elements.

In the context of the TFPSN, and following this pioneering work of Lappas et al. (2009), a vast literature has emerged where specific approximation algorithms are developed for specific formulations of the problem. The problem has been formulated in different ways depending, for instance, on the definition of the communication costs between experts, capacity or other constraints. As a result, numerous approximation algorithms have been proposed. In general, the processing of existing methods can be divided into two phases: take experts as search target, search the experts having the required skills as candidates firstly, and then select team members from the candidates based on their constraints. The constraints are used for measuring the effectiveness of teams. The constraints explored in the literature are based on the shortest path distances, density, and the cost of the minimum spanning tree of the subgraph induced by the team. The shortcoming of the two-phase algorithms is relatively poor time efficiency and their specificity to the problem at hand.

As discussed in Bousaïd et al. (2013), an alternative way of effectively tackling the team formation problem is through metaheuristic methods. These are general-purpose algorithms that can be used to solve almost any optimization problem. Their global search capabilities, their flexibility, robust performance and adaptability are all considered as outstanding characteristics of these methods when searching for optimal solutions. Despite these advantages, there is considerable less amount of research applied in the context of the TFPSN. The present work aims at reducing this gap by applying to this problem state-of-the-art methods which are inspired on natural computation.

1.2.2 Natural Computing Approach

Natural computing has established itself as a prominent paradigm in the domain of general purpose search and optimisation methods. Brabazon et al. (2015) survey the main developments in this field and note that, as our understanding of natural phenomena has deepened, so has our recognition that many mechanisms in the natural world parallel computational processes and can therefore serve as an inspiration for the design of problem-solving algorithms (defined simply as a set of instructions for solving a problem of interest). Within the field of natural computing, Evolutionary Algorithms (EAs) are considered both by researchers and practitioners as effective optimisers. EAs have as their objective to mimic processes from natural evolution, where the main concept is survival of the fittest. Their applications range from academic research and industrial engineering to financial analysis and even art production. EAs define practical and robust optimization and search methodologies.

The idea of a computer-simulated evolutionary process dates to the very dawn of digital computing, being introduced in the writings of Alan Turing in (1948, 1950). Originating from more traditional approaches to problem solving, EAs then developed into diverse branches. As described in Engelbrecht (2007), the main branches of this field are evolution strategies, evolutionary programming, genetic programming, differential evolution and genetic algorithms. The latter originate from the pioneer work of Holland (1975). As described in detail in Rowe (2015), these algorithms are inspired by concepts in evolutionary biology such as inheritance, mutation, selection, and crossover, in which a population of random chromosomes represents many possible solutions to the problem to be solved.

Before converging to some optimum in the search space, EAs must evaluate an adequate number of candidate solutions whereby the low-fitness individuals are discarded and the high-fitness candidates are selected to generate new offspring. Therefore, an EA solving real-world problems with polynomial complexity can be impractical due to costly fitness (function) evaluations. While EAs are effective at finding acceptable solutions, they need to be coupled with efficiency enhancement techniques to be competitive with other conventional optimizers. Conveniently, a large corpus of efficiency enhancement techniques exists for EAs, as discussed, for instance, in Luong et al. (2012).

Another important aspect with EAs is the need to preserve the diversity of the population and thus avoid getting stuck at local optima. The trade-off between exploration and exploitation is a central theme in this field. EAs are characterized by the representation of the individual, the evaluation function representing the fitness level of the individuals, and the population dynamics such as population size, variation operators, parent selection, reproduction and inheritance, survival competition method. Crepinsek et al. (2013) note that, to have a good balance between exploration and exploitation, these components should be designed properly.

1.2.3 Quantum-Inspired Approach

One line of research has attempted to improve the performance of EAs through the incorporation of quantum principles. This has led to the so-called quantum evolutionary algorithms (QEAs). Manju et al. (2014) provide a detailed review of applications within the field of computational intelligence, whereas Zhang (2011) presents a survey and empirical study of QEAs. This can be thought of as the result of combining the worlds of biological evolution and quantum systems. The term "inspired" must be stressed as these are algorithms that run on classical computers; interestingly, the use of computational paradigms with proximity to the quantum concepts will require less translation to quantum machine language when the quantum computers become available. [2]

QEAs can be regarded as a probabilistic system in which the probabilities related to each state are used to describe the behaviour of the system. The first application of these concepts was proposed by Narayanan and Moore (1996). The authors solved the traveling salesman problem in which the crossover operation was performed based on the concept of interference.[3]

QEAs work with a quantum population. Each chromosome in the quantum population is represented as a string of quantum bits, also called as qubits, with the advantage that it can represent a linear superposition of states in the search space probabilistically. Superposition enables a quantum chromosome to store exponentially more data than a classical chromosome of the same size. Further, whereas an operation applied to a classical chromosome produces one result, in a quantum-inspired algorithm, an operation applied to the quantum chromosome produces a superposition of all possible results. Thus, the implicit parallelism in QEAs leads to better population diversity than in the classical representation.

QEAs have been applied in multiple domains, particularly to solve difficult combinatorial optimisation problems. Han and Kim (2000, 2002) were the first to implement a QEA that works with a quantum population comprising of binary observation quantum chromosomes with each chromosome represented as a string of qubits. The authors investigated the applicability of QEAs to the well-known difficult combinatorial optimization problem, the 0-

[2] Yingchareonthawornchai et al. (2012) present a first attempt to programming a genetic algorithm in a quantum computer.

[3] Du and Swamy (2016) provide a description of the main quantum concepts such as interference and superposition.

1 knapsack problem. This has been a benchmark problem widely used to test the properties of different QEAs as discussed in Patvardhan et al. (2016).

There is ample evidence that quantum-inspired evolutionary algorithms display better convergence and search abilities than the classical evolutionary methods, giving better solutions in shorter time. They attribute the global search ability of QEAs to the diversity generated by the intrinsic probabilistic representation. As discussed in Platel el at (2009), QEAs can be regarded as a kind of Estimation of Distribution Algorithm. These algorithms replace the traditional reproduction mechanism of EAs, i.e. genetic operators, with probabilistic model estimation, followed by sampling individuals from the estimated model. Hauschild and Pelikan (2011) provide a survey of this field of research.

Another important feature of QEAs is their ability to explore the search space even with a small number of individuals and achieve a proper balance between exploration and exploitation. However, and despite their numerous advantages, QEAs face some shortcomings common to many EAs. Improving the performance of these algorithms has thus been an active field of research as evidenced by the work of Tarayani and Akbarzadeh (2014).

1.3 Contribution of this Work

As discussed in the previous section, one of the main characteristics of QEAs is their ability to produce an adequate balance between intensification and diversification during the search process. This property helps to deal with difficult problems like the one considered here, in which there are usually many local optima.

The main contribution of the present work is the application and evaluation of methods based on QEAs to the TFPSN. Specifically, a QEA is applied and contrasted to a canonical genetic algorithm. A series of numerical experiments based on synthetic data is carried out to assess the suitability of the proposed method. The results show that the QEA is able to significantly improve the quality of the solution when compared to a standard genetic algorithm.

1.4 Organisation of the Thesis

The present thesis is organised as follows. Chapter 2 provides a formal definition of the problem object of study. Chapter 3 then presents a detailed review of the literature on methods for tackling the TFPSN. Chapter 4 describes the QEA that is used to solve the TFPSN. Chapter 5 presents the results of a series of numerical experiments and tests aimed at evaluating the performance of the QEA described in the previous chapter. Finally, Chapter 6 provides some concluding remarks and possible extensions of the work described in this thesis.

Chapter 2 Team Formation Problem in Social Networks

This chapter provides a formal definition of the Team Formation Problem in Social Networks (TFPSN). Essentially, given a social network of experts and a collaborative task that requires a set of skills, the team formation problem aims at finding a team of experts who can cover all the required skills and communicate with one another in an effective manner.[4]

2.1 Intuitive Description

To illustrate the problem, we borrow the example discussed in Wang et al. (2015). The authors consider a social network represented by a graph as the one shown in Figure 2.1. The graph, denoted as $G(V,E)$, is composed of a set V of nodes (experts) and a set E representing the links between the experts. In the example considered here, the network is formed by six experts denoted as $V = \{v_1, \dots, v_6\}$. Each expert is associated with a specified set of skills. For instance, expert four, v_4, has two skills which are denoted as s_4 and s_5 respectively.

The edges of the graph have an associated value or weight that represents a measure of the collaboration between the experts. This has different interpretations depending on the specific problem considered. For instance, Lappas et al. (2009) define these weights as communication costs between experts. In this way, a low-weight edge between nodes v_i and v_j implies that the two experts, v_i and v_j, can collaborate or communicate more easily than candidates connected with a high-weight edge. These weights can be instantiated in different ways in different application domains. For example, in a company, the weight between two employees may correlate to the length of the path from one employee the other through the organizational chart. In a scientific research community, the weight between two scientists is related to the total number of publications they have co-authored. Interpersonal relationships among individuals can also be used to calculate the weights.

The collaboration cost between two experts, v_i and v_j, is represented as the weight of edge (v_i,v_j). Note that this cost is not necessarily calculated along the shortest path between the

[4] It must be emphasised that the notion of communication is wide and generally reflects the degree in which two experts can collaborate effectively. Hence, these two terms, communication and collaboration, will be used interchangeably.

experts, that is, along the path between them with a minimum number of edges. For example, the collaboration cost between experts v_1 and v_3 is 0.2 in Figure 2.1; however, if the cost between expert 1 and expert 3 was higher than 0.4, it would be preferable to follow the route via expert 2, as it would be cheaper than the direct one.

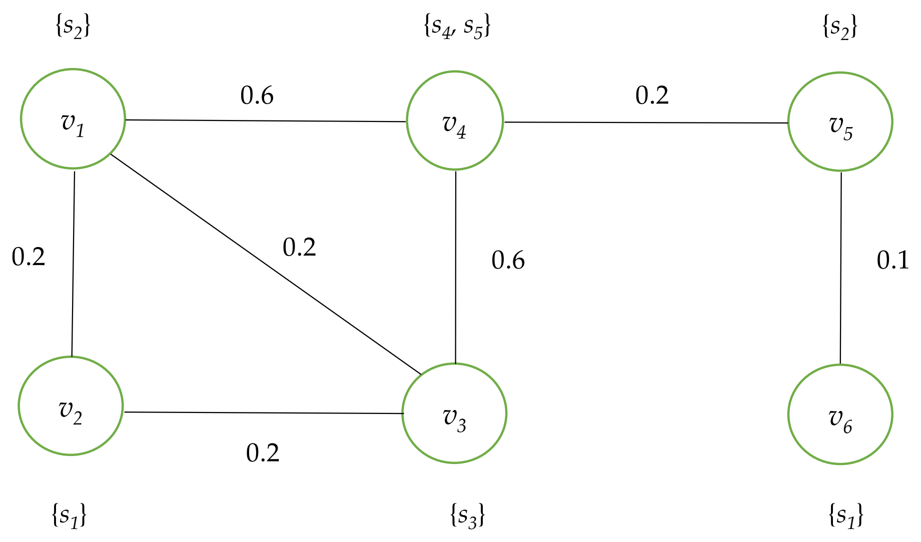


Figure 2.1 A Social network of experts and their collaboration costs

Consider the problem in which, given a task T with five required skills $\{s_1, s_2, s_3, s_4, s_5\}$, the aim is to find a team of experts, denoted by $X \subseteq V$, that collectively cover all the required skills with the least collaboration cost. In this problem, two of the teams that cover all the required skills are $X_1 = \{v_1, v_2, v_3, v_4\}$ and $X_2 = \{v_3, v_4, v_5, v_6\}$ (see Figure 2.1). However, the solution provided by the algorithms will ultimately depend on the cost function specified for a team.

In the literature, two popular metrics of collaboration cost for a team are the *diameter distance* and the *Steiner distance*. These will be formally defined later in this chapter, but the former is the maximum collaboration cost among the cheapest paths between any two experts in the team, whereas the latter is the cost of the minimum spanning tree for the subgraph corresponding to the chosen team, that is, the sum of the collaboration costs of all the edges in the mentioned minimum spanning tree.[5]

One can observe that the collaboration cost of team X_1 under the *diameter* distance is 0.8, since this is the maximum among the cheapest costs between any two experts in X_1, which is

[5] A spanning tree of an undirected graph is a subgraph that is a tree (any two vertices are connected by exactly one path) and contains all the nodes of the undirected graph. In a weighted graph, a minimum spanning tree is a spanning tree with minimum weight, where the weight of a tree is calculated as the sum of its edges' weights.

associated to the experts v_2 and v_4 in this case: the shortest path is $\{(v_2,v_3), (v_3,v_4)\}$ and the corresponding cost is 0.2 + 0.6 = 0.8.

The Minimum Spanning Tree of the graph representing team X_1 in Figure 2.2 is given by $\{(v_1,v_2), (v_1,v_3), (v_3,v_4)\}$, resulting in a cost under the *Steiner* distance of 1 = 0.2 + 0.2 + 0.6.

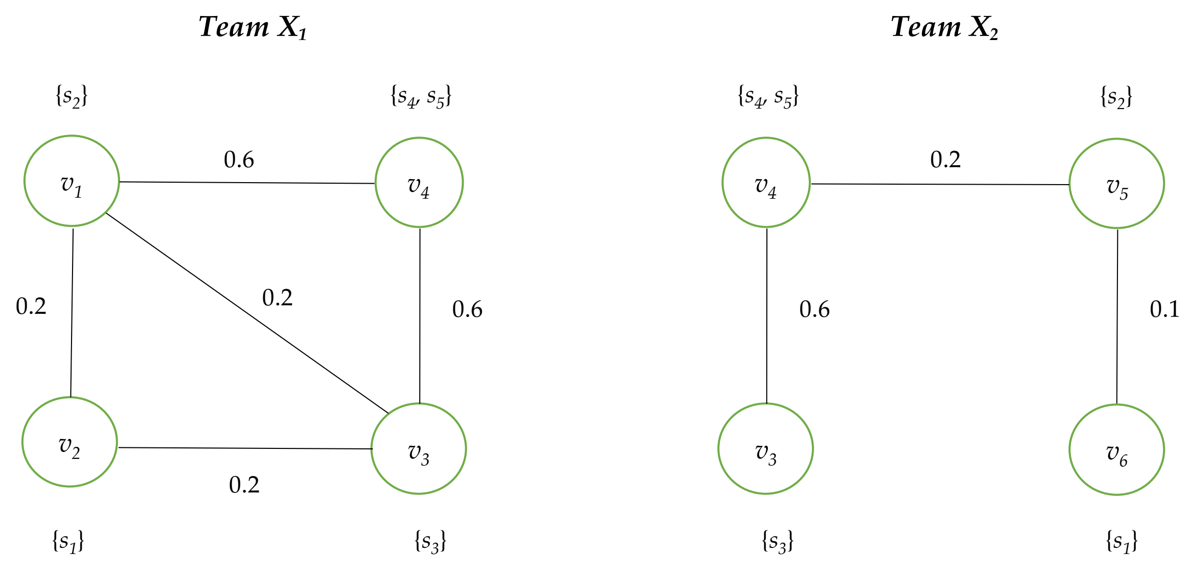


Figure 2.2 Two feasible teams for the task T= {s_1, s_2, s_3, s_4, s_5}

The collaboration cost of team X_2 under the *diameter* and *Steiner* distances is 0.9 = 0.6 + 0.2 + 0.1 corresponding to the subgraph $\{(v_3,v_4), (v_4,v_5), (v_5,v_6)\}$. Hence, team X_1 has the lower collaboration cost according to the *diameter* distance, while team X_2 has the lower collaboration cost according to the *Steiner* distance. It is clear thus that, in the formulation of the TFPSN, the way the cost function is defined is critical. There is an ample literature on this subject as reviewed in Chapter 3.

As noted above, the collaboration (communication) cost associated to the link between any two experts v_i and v_j, denoted by $e_{ij} \in E$, can be defined in different ways depending on the nature of the problem. One possible definition is given by the following expression:

$$e_{ij} = 1 - \frac{|s(v_i) \cap s(v_j)|}{|s(v_i) \cup s(v_j)|}$$

where $s(v_i)$ represents the set of skills of expert v_i. Formally, the weights on the edges represent pairwise *Jaccard* distances between the skills of all pairs of connected nodes. The intuition is, for instance in a network of scientist co-authoring papers, to represent how many papers both authors have written together out of all the papers published by the authors.

2.2 Formal Definition

To formalise the TFPSN, we first introduce some notation and definitions. There is a collection of N candidate experts, $V = \{v_1, \dots, v_N\}$ and a set of M specified skills $S = \{s_1, \dots, s_M\}$. Each expert v_i is associated with a set of skills $s(v_i) \subseteq S$. The set of experts having skill s_j is denoted as $C(s_j)$, where $C(s_j) \subseteq V$. A task T is modelled by a set of required skills, that is, $T = \{s_{T(1)}, \dots, s_{T(k)}\} \subseteq S$. A team of experts $X \subseteq V$ is said to be *feasible* for task T, if and only if the experts in X collectively cover all the required skills, that is, $T \subseteq \cup_{v_i \in X} s(v_i)$.

The collaboration cost is represented through the closeness of the experts in a social network $G(V,E)$. Without loss of generality, the graph G can be assumed to be connected as it is always possible to transform any subgraph to a connected one where there is a path between every pair of vertices.[6] The collaboration cost of two experts, v_i and v_j, is given by the cheapest path between them, denoted as $sp(v_i,v_j)$. Table 2.1 summarises the notation for this problem.

Notation	Definition
V	A set of experts
$G(V,E)$	Experts social network
S	Set of skills
T	A task with required skills
X	Experts team
$C(s_i)$	A set of experts skilled in s_i
$s(v_i)$	Skills of expert v_i
e_{ij}	Edge between experts v_i and v_j
$sp(v_i,v_j)$	Cheapest cost between experts v_i and v_j

Table 2-1 Notation for the TFPSN

[6] In this case, by simply adding edges with a very high weight between every pair of nodes belonging to different connected components. Note that this very high weight is a number higher than the sum of all pairwise cheapest paths in G.

It is now possible to formally define the TFPSN as follows: Given a social network of experts denoted by the undirected weighted graph $G(V,E)$ and where the experts are associated with specified skills in S, the TFPSN is to find the team of experts $X \subseteq V$ that can collectively cover all the required skills for task T with the lowest collaboration cost.

The problem statement above is relatively general and hence does not explicitly mention the form of the collaboration cost function. The following are examples of this cost function which are widely used in the literature:

<u>*Diameter Distance:*</u> is defined as the maximum collaboration cost among the cheapest paths between any two experts in the team X, that is,

$$Cc\text{-}D(X) = \max_{v_\mathrm{i}, v_\mathrm{j} \in \mathrm{X}} sp\left(v_\mathrm{i}, v_\mathrm{j}\right).$$

<u>*Steiner Distance:*</u> is denoted as $Cc\text{-}ST(X)$ and represents the weight cost of the minimum spanning tree for the subgraph $G' \subseteq G$ formed by team X.

<u>*Sum of Distances:*</u> is defined as the sum of all cheapest paths between any two experts in team X, that is,

$$Cc\text{-}SD(X) = \sum_{v_i, v_j \in X} sp\left(v_i, v_j\right).$$

From a practical point of view, the TFPSN can be solved in two steps. First, a series of P teams of experts, corresponding to feasible solutions, is obtained. The collection of these sets is denoted by $\boldsymbol{X}=\{X_1,\ldots,Xp\}$. The next step consists of checking which one of the sets in $\boldsymbol{X}$ minimizes the collaboration cost. That is,

$$X^* = arg \min_{X_i \in \boldsymbol{X}} Cc(X_i)$$

subject to

$$\forall s_i \in T, \exists C(s_i) \notin \emptyset$$

The candidate solutions for this problem can be encoded in a binary form, so that its representation is adequate for the application of genetic algorithms, as shown in detail in Chapter 4.

Lappas et al. (2009) showed that the TFPSN is NP-hard. They also showed that this is an instance of the *set cover problem*. This problem is one of the classical problems in complexity theory and computer science.[7] This is regarded as one of the most important discrete optimization problems because it can be formulated as a model for various real-life problems, such as vehicle routing, resource allocation, scheduling problem, or facility location problem.

Given that the problem is proved to be NP-hard, approximate algorithms need to be considered. For instance, Beasley et al. (1996) present a genetic algorithm to tackle this problem.

[7] Formally, given a group of sets, denoted by N, let X be the union of all the sets. An element is covered by a set if the element is in the set. A cover of X is a group of sets from N such that every element of X is covered by at least one set in the group. The *set cover problem* is to find a cover of X of minimum size. For instance, let N={{1,2,3}, {2,3}, {3,4}, {4,5}}. The union of these sets is X={1,2,3,4,5} which can be achieved by the set {{1,2,3}, {4,5}} whose size is 2.

Chapter 3 Review of the Literature

The TFPSN belongs to a class of complex combinatorial problems that require the use of computational methods that cannot in general guarantee the optimality of the solution as exhaustive search is not feasible. As a result, algorithms that provide a reasonable good solution are required. These inexact (or sub-optimal) methods can be subdivided into two categories: approximation algorithms and metaheuristic methods.

As shown in Figure 3.1, on the one hand, there is a set of algorithms, referred to as approximation algorithms, that provide "close" to optimal solutions. These are efficient deterministic algorithms that find reasonable solutions to NP-hard problems, such as the TFPSN, with provable guarantees on the distance of the returned solution to the optimal one. One approach to design approximation algorithms is through greedy methods that make the choice that seems to be the best at each step. This means that a locally-optimal choice is made in the hope that this choice will lead to a globally-optimal solution, which is not always the case. Section 3.1 presents some approximation algorithms applied in the context of the TFPSN.

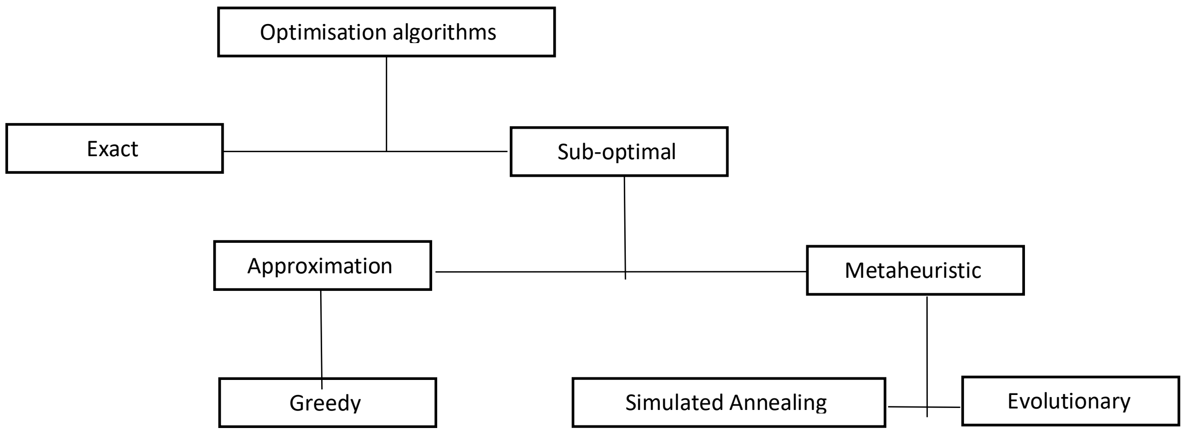


Figure 3.1 Classification of optimisation algorithms

Section 3.2 focus on metaheuristic methods. A metaheuristic is a general algorithmic framework which can be applied to different optimisation problems with relatively few modifications to make them adapted to a specific problem. These algorithms use a certain trade-off of randomization and local search. Quality solutions to difficult optimisation problems can be found in a reasonable amount of time, but there is no guarantee that optimal solutions can be reached. Some of the most prominent metaheuristic algorithms are inspired in nature, such as evolutionary algorithms or simulated annealing. The former constitute examples of global, population-based algorithms, whereas the latter illustrates a single-point local solution method.

3.1 Review of Approximation Methods

The discussion so far has put forward the importance of studying the formation of groups of agents or teams considering the relationship amongst the members of the teams. The question is how to form those teams in an optimal manner. The response to this question will depend on what the specific objective to achieve is, specifically, how the model is formulated.

The literature on the TFPSN has grown significantly in recent times, giving rise to several approximate algorithms that are able to tackle the problem effectively. Each formulation focuses only on a subset of design criteria such as skill coverage, social compatibility, economy, skill redundancy, etc. Depending on the specific formulation of the problem, a cost function is defined. This cost function could be defined relative to different elements, such as, collaboration costs, personal costs, number of skilled experts, capacity constraints, balance of workload, team leadership, skilfulness, or amount of tasks performed. Wang et al. (2015) compare state-of-the-art team formation algorithms. They propose a benchmark that enables fair comparison amongst these algorithms and then implement these algorithms using a common platform and evaluate their performance using several real datasets.

Lappas et al. (2009) were the first to consider the team formation problem in the presence of a social network of individuals. They explicitly introduced communication costs and studied two instances of this problem depending on the way those costs were considered: the *diameter* distance and the *minimum spanning tree* communication cost. They addressed the problem of forming a team of skilled individuals to perform a given task, while minimising the communication cost among the members of the team. The authors analysed the problem rigorously by showing that, in both instances of the cost function, it is NP-hard. They presented accordingly approximation algorithms for the solution of the problem. The authors also carried out experiments and illustrated that their problem definitions, as well as the algorithms, work well in practice and give useful and intuitive results.

Li et al. (2015) extend the original team formation problem to a generalised version in which the number of experts selected for each required skill is also specified. The proposed teams need to contain an adequate number of experts for each required skill. The authors develop two approaches for this problem. Firstly, they devise a density-based seed selection strategy as well as a novel grouping-based method to compose the team for generalised tasks.

Rangapuram et al. (2015) generalised even further the TFPSN to accommodate realistic scenarios. In particular, these authors considered the inclusion of a pre-designated group of experts, restrictions in the size of the team as well as incompatibility constraints between experts.

The issue of the team leader has also been considered in the definition of the TFPSN. Kargar et al. (2011) introduced the problem of finding a team of experts with a leader who is responsible for monitoring and co-ordinating the project, and thus a different communication cost function was used. To solve this problem, an exact polynomial algorithm was proposed. They showed that the total number of teams may be exponential with respect to the number of required skills. Teng et al. (2014) consider the problem of choosing multiple leaders organised into a hierarchy where each leader is responsible for only a limited number of team members. In this paper, the authors propose the team formation problem with the communication load constraint in social networks. The communication load constraint limits the number of team members a leader communicates with. More recently, Shi et al. (2017) argue that, as the number of required skills increases and the team grows, a single leader is not sufficiently capable of administering a large project, since the leader may not have enough time to communicate with all the team members. Therefore, the team would be divided into smaller sub-teams, each of them with a leader. They propose three basic algorithms, and explore a self-organising mechanism to determine the role of everyone in a team (as a leader or as a member).

Another topic analysed in the literature is the consideration of personnel costs. These are defined as the number of skills the expert is responsible for in the project. The more skills the expert has in the project, the more responsibility he or she has and, as a result, the higher is the associated cost. In a series of papers, Kargar et al. (2012) and Kargar et al. (2013a, 2013b) studied the problem of finding an affordable and collaborative team from an expert network that minimises two objectives: the communication cost amongst team members and the personnel cost. They propose an approximation algorithm based on converting the input graph with both node and edge weights into a graph with weights on edges only. The methods the authors propose are based on either iteratively replacing cheapest experts with more expensive ones to improve the combined cost or incrementally adding experts with minimum cost contribution.

Other formulations include capacity constraints. In this case, for a given task, the goal is to find a team of users from the social network who are socially effective in terms of collaboration and a division of the task among team members so that no user is overloaded by the assignment, that is, no user is assigned task beyond her capacity. Datta et al. (2012) presented several algorithms to solve the TFPSN taking into consideration these type of capacity constraints. They presented an approach that attempts to find teams that are socially close as well as division of the task so that no user is overloaded by the assignment. Anagnostopoulos et al. (2012) studied a version of the TFPSN considering both the team communication overhead and workload balancing.

Time limits have also been considered when formulating the TFPSN. In this case, the objective is to find a team that not only covers the skills that the task requires but also that completes the different tasks in the specified time. For instance, Yang and Hu (2013) propose a standard to measure the cost of team building containing both the communication cost and the time. They present two approximation algorithms to solve the problem and provide experimental evidence based on real datasets showing satisfactory results.

Recently, Fathian et al. (2018) have proposed an optimization model for the formation of a reliable team of experts, who have a certain number of skills and form part of a collaboration network. The proposed mathematical model maximized team reliability by considering the probability of unreliable experts that may leave the team and proposed a backup for each unreliable member.

3.2 Review of Metaheuristic Methods

The approximation methods discussed in Section 3.1 tend to be problem specific and thus cannot easily be applied to alternative specifications of the problem. Hence, it would be desirable to consider other more wide-ranging methods. As discussed in Chapter 2, the TFPSN is a difficult combinatorial optimisation problem. Hence, the use of metaheuristic algorithms constitutes an alternative approach to solving this problem.

3.2.1 Evolutionary Algorithms

Evolutionary algorithms, as described in Bäck et al. (2000), Eiben et al. (2003) and de Jong (2006), are a class of metaheuristic methods which have been shown to be powerful optimisation techniques that can be applied to an ample spectrum of cases. Moreover, the team formation problem becomes much more challenging in a large social network. With a linear increase in experts, the number of combinations to check grows exponentially. At the same time, the best connected experts, which are usually also the best skilled ones, display an increasing number of relations. Consequently, checking for recommendations becomes computationally expensive.

In this section, we review some of the main references in the literature that have applied metaheuristic evolutionary methods. In this context, the team formation problem is defined as finding the best match of experts to required skills considering multiple dimensions from technical skills, cognitive properties and personal motivation. These papers largely focus on properties of individual experts that are independent of the resulting team configuration. For instance, Agustin-Blas et al. (2011) considered a genetic algorithm (GA) to tackle the team formation problem. Their focus was on solving grouping problems in which several items had to be assigned to a set of predefined groups or teams. Interestingly, they consider a variable-length encoding approach which is particularly suitable for this type of problems. To enhance population diversity, they consider an island model.[8] A similar evolutionary approach was used by Strnad and Guid (2010). In this case, the objective of their model is to produce a group of teams which maximizes the expected utilization of available technical and functional capacities described by fuzzy values, that is, vague or imprecise inputs. This is particularly appealing in contexts where the problem statement and expert requirements are uncertain such as R&D activities. Pitchai et al. (2016) developed a Quantum-based GA to identify near optimal teams that optimises a fuzzy criterion, obtained from initial team requirements. The efficiency of the proposed design was tested on a variety of artificially constructed instances. The authors showed that results of the proposed optimisation algorithm are practical and effective.

Probably, one of the first papers to explicitly consider social interactions within the team formation problem is Dorn et al. (2011). The authors apply GAs and simulated annealing for determining effective workforces which provide sufficient skill coverage while achieving adequate team connectivity.[9] The authors also show that such team configurations promise higher probability of success in future collaborations compared to team arrangements that neglect social relations and account for individual skills only. The authors consider load constraints that determine the short-term availability of users. Experts that are not available cannot become part of a new team. Instead of eliminating them from the candidate set, the authors let them act as referees by applying their social network for recommending other experts of their respective fields. The use of implicit recommendations of collaboration partners help to deal with sparsely connected networks.

Awal and Bharadwaj (2014) addressed the TFPSN by proposing a novel way to quantify and optimise a team's collective ability to solve a specific task. They presented a GA that maximises the collective abilities of teams. This is defined as an aggregate measure of

[8] Island Models are a popular and efficient way to implement a genetic algorithm on both serial and parallel machines. In a parallel implementation of an Island Model each machine executes a genetic algorithm and maintains its own subpopulation for search. The machines work in consort by periodically exchanging a portion of their population in a process called migration. See Whitley et al. (1999) for a detailed exposition.

[9] Simulated annealing is an optimisation method inspired by the physical process of heating a material and then slowly lowering the temperature to decrease defects. The extent of the search is based on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective, but also, with a certain probability, points that raise the objective in order to avoid local optima.

collective intelligence, rather than the individual abilities of the team members. The authors quantify this measure by introducing a collective intelligence index for the team. This index is defined as a linear combination of an expertise score and a trust-based collaboration score. A series of experimental results reveal that the collective intelligence index increases as the sparsity level of the graph decreases, indicating that well-connected experts are more suited to achieving successful collaborations and may enhance the learning capabilities of the experts through knowledge sharing.

In a subsequent paper, Awal and Bharadwaj (2018b) considered the TFPSN in more realistic settings where the team composition must satisfy certain constraints. The authors show that, in some cases, only a few suitable experts having high reputation in the team is sufficient to complete the task. The authors also showed that not all experts having high reputation or expertise are always needed to form a successful team. They proposed a GA to study the problem and introduced risk estimation strategies to determine the suitability of a team for a given task. The experimental results establish that their proposed model is useful in practical scenarios and discovers more coherent and collectively intelligent teams having low inherent risks.

Other relevant contributions include Zamudio et al. (2016) who apply a GA to select committees with independent members in social networks. Pendharkar (2013) proposes a GA-based intelligent agent that learns a team member's preferences from past actions, and develops a team co-ordination schedule by minimising schedule conflicts between different members serving on a virtual team. Esgario et al. (2019) proposed an evolutionary approach to solve the Team Formation Problem based on sociometry, which is a qualitative method for measuring social relationships in order to form teams with high cohesion in a reduced time. The genetic operators were adapted to deal with the constraints of the problem. In addition, the penalty method was used to force the algorithm to find feasible solutions.

3.2.2 Other Metaheuristic Methods

The discussion so far has focused on application of evolutionary algorithms to solve the TFPSN. However, it is worthwhile commenting on other metaheuristic approaches that have recently been applied in this context. El-Asmawi et al. (2018), for instance, have proposed a particle swarm optimization algorithm (PSO) for solving the TFPSN. These algorithms are inspired by the social foraging behaviour of some animals such as the flocking behaviour of birds and the schooling behaviour of fish. PSO shares many similarities with evolutionary computation techniques such as GAs. The system is initialised with a population of random solutions and searches for optima by updating generations. The goal of the algorithm is to locate the optima in a multi-dimensional space.[10] The authors modified the standard PSO algorithm to ensure the consistency of the capabilities and the skills to perform the required project. The results presented by the authors indicate that the algorithm could obtain promising solutions to difficult instances of the problem in a reasonable timeframe.

Basiri et al. (2017) proposed a swarm-based algorithm which simulates the brain drain phenomenon. Brain drain refers to the international transfer of human capital and considerably applies to the migration of highly educated persons from developing to developed countries. The initial population is divided into some collections which are called countries. There is a social network in each country which specifies each individual neighbour. The individuals try to discover increasingly better regions of the search space. An unhappy individual which has not had any progress during previous iterations, will try to change its current position and go somewhere else better than the source country. This migration affects the future migration flow of the individual's neighbours within the social network. The results presented by the authors demonstrate the effectiveness and superiority of the proposed algorithm in comparison with the standard evolutionary algorithms.

[10] Specifically, the algorithm is executed like a simulation, advancing the position of each particle in turn based on its velocity, the best known global position in the problem space and the best position known to a particle. Over time, through a combination of exploration and exploitation of known good positions in the search space, the particles cluster or converge together around an optimum.

Chapter 4 Design of a Quantum Evolutionary Algorithm for the Team Formation Problem in Social Networks

This chapter formally introduces the evolutionary algorithm that we use to solve the TFPSN. In order to provide a context to the exposition, the first section introduces the common elements that characterise evolutionary algorithms in general. The second section presents in some detail the canonical version of QEAs. The main concepts and notation are described first, whereas a step-by-step description of the algorithm is given next. The third section describes how the TFPSN can be addressed within the framework of the canonical QEA.

4.1 Evolutionary Algorithm

Evolution is to be understood as an optimisation process where the aim is to improve the ability of a system to survive in dynamically changing and competitive environments. Evolutionary processes represent an archetype whose application transcends their biological root. These processes can be distinguished by means of four key characteristics, which are:

i. a population of entities;

ii. mechanisms for selection;

iii. the generation of variety; and

iv. retention of fit forms.

In biological evolution, species are positively or negatively selected depending on their relative success in surviving and reproducing in the environment. Differential survival, and variety generation during reproduction, provide the engine for evolution, as illustrated in Figure 4.1.

The selection step is a pivotal driver of EAs. The step is biased to preferentially select better (or fitter) members of the current population. The generation of new individuals creates new members which bear some similarity to their predecessors but are not identical to them. Hence, each individual represents a trial solution in the environment, with better individuals having increased chance of influencing the composition of individuals in future generations.

This can be considered as a search process, where the objective is to continually improve the quality of individuals in the population.

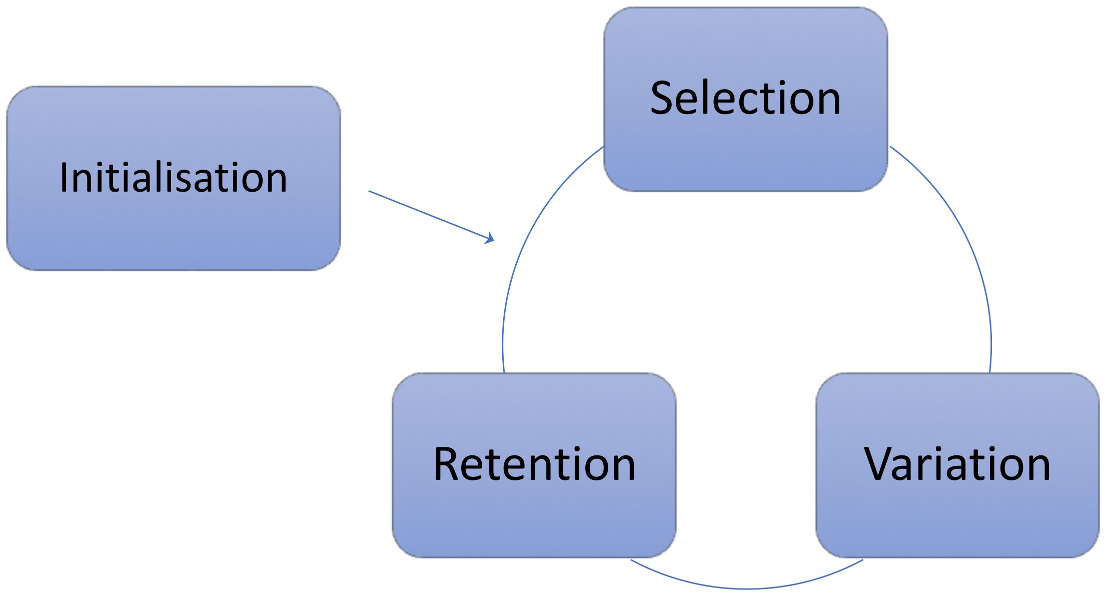


Figure 4.1 The evolutionary cycle

EAs can thus be broadly characterized formally as follows:

$$x[t+1] = r\left(v\big(s(x[t])\big)\right),$$

where $x[t]$ is the population of encodings at time step t, $s(.)$ is the selection operator for mating, $v(.)$ is the random variation operator (crossover and mutation) and $r(.)$ is the replacement selection operator.

An important step in the design of an EA is to find an appropriate representation of candidate solutions. The efficiency and complexity of the search algorithm greatly depends on the representation scheme. In genetics, a strong distinction is drawn between the genotype and the phenotype; the former contains genetic information, whereas the latter is the physical manifestation of this information. Both play a role in evolution as the biological processes of diversity generation act on the genotype, while the 'worth' or fitness of this genotype in the environment depends on the survival and reproductive success of its corresponding phenotype.

The classical representation scheme for GAs is binary vectors of fixed length. In the case of an n-dimensional search space, each individual consists of n variables with each variable encoded as a bit string. If variables have binary values, the length of each chromosome is n bits.

Inspired by the concept of quantum computing, QEAs are designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process. Like conventional evolutionary algorithms, QEAs are characterised by the representation of individuals, population diversity, and the use of a fitness evaluation mechanism.

Unlike the conventional framework for evolutionary algorithms, QEAs describe individuals through a Q-bit representation. By employing an observation process, the algorithm connects the Q-bit representation with the candidate solutions. Another key element is the use of the so-called Q-gates as an evolutionary operator to obtain fitter individuals.

4.2 The Quantum Evolutionary Algorithm

This section describes the canonical QEA introduced by Han and Kim (2002). We first focus on the Q-bit representation, and then the main elements of the algorithm are explained. In order to describe the QEA, it might be useful to introduce some basic concepts of quantum computing.

The smallest unit of information stored in a two-state quantum computer is called a *quantum bit* or *qubit*.[11] This may be in the "0" state, in the "1" state, or in any superposition of the two. The state of a qubit can be represented using the so-called *ket* notation as:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are the real and complex components of the complex number $\alpha + \beta i$, whose modulus is equal to 1, and that determines the probability amplitudes of the corresponding states.

The value $|\alpha|^2$ gives the probability that the qubit will be found in the "0" state, and $|\beta|^2$ gives the probability that the qubit will be found in the "1" state. Normalization of the state to unity guarantees that $|\alpha|^2 + |\beta|^2 = 1$. The state of a qubit can be changed by the operation with a

[11] In the quantum computing literature, the basic unit of information is referred to as *qubit*, whereas in the evolutionary computing literature it is represented as *Q-bit*. The latter will be used in this thesis.

quantum gate. As shown later, this is an important element that will influence the search process in the optimization algorithm.

4.2.1 Quantum-bit Representation

In a QEA, encoding the solutions onto chromosomes is based on the probabilistic Q-bit representation. Formally, a Q-bit is defined as a pair of real numbers (α, β) written as a column vector:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}.$$

A Q-bit may be in the "0" state, in the "1" state, or in a linear superposition of the two. A string of Q-bits gives rise to a Q-bit individual. This is defined as:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix},$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1$ for $i = 1,2,\dots,m$. The Q-bit representation has the advantage that it can implement a linear superposition of states.

To illustrate these ideas, let us consider the context of the TFPSN. Assume that we have three experts, hence $m = 3$. Each expert can or cannot be chosen to participate in the team. If an expert is chosen to form part of the team, it is assigned a value of "1", whereas it is assigned "0" in the case of not being chosen. In this example, an individual (set of three experts) has $2^3 = 8$ possible values:

$$|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle.$$

The first value (or state) encodes a solution where no expert is chosen, while in the last one all three experts are selected to participate in the team. Next, let us assume that there is a Q-bit with three pairs of amplitudes such as:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{bmatrix}.$$

The case $|000\rangle$ has an amplitude coefficient given by the product $\alpha_1\alpha_2\alpha_3$ which, according the values above, is equal to $\frac{1}{4} = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\frac{1}{2}$.

The case $|011\rangle$ has an amplitude coefficient defined by the product of $\alpha_1\beta_2\beta_3$, which is equal to $\frac{-\sqrt{3}}{4} = \frac{1}{\sqrt{2}}\frac{-1}{\sqrt{2}}\frac{\sqrt{3}}{2}$. A similar process can be followed for the other possible cases. Altogether, the individual can be represented as:

$$|\Psi_3\rangle = \frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \frac{1}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle$$

In the process of observation, each of the eight states can be selected with an associated probability. These probabilities are given by the square value of the amplitude coefficients. Respectively, these are equal to:

$$\frac{1}{16}, \frac{3}{16}, \frac{1}{16}, \frac{3}{16}, \frac{1}{16}, \frac{3}{16}, \frac{1}{16}, \frac{3}{16}.$$

Evolutionary computing with Q-bit representation is characterised by featuring more population diversity than other representations, since it can represent a linear superposition of states probabilistically. While only one Q-bit individual is enough to represent eight states, in binary representation an individual can only represent one of the possible eight states:

(000), (001), (010), (011), (100), (101), (110), and (111).

The Q-bit representation differs from others considered, for instance, in genetic algorithms, such as binary, integer or floating representations. Figure 4.2 provides an illustration in this regard.

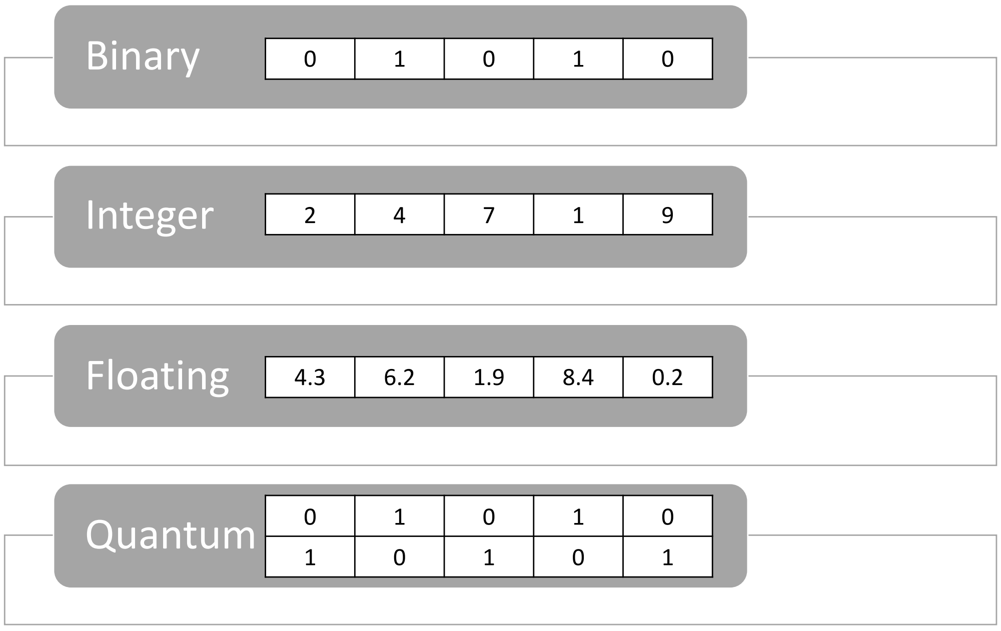


Figure 4.2 Different types of representations for individuals in GAs

4.2.2 Main Components of QEAs

Before presenting the details of the QEA designed in the present work, it would be helpful to provide an overview of the main components. This is illustrated in Figure 4.3, that shows the pseudo-code for the canonical version of the algorithm.

The first step corresponds to the initialisation phase. A set of Q-bit individuals, denoted by $Q(t)$, is thus generated, usually by means of a random generating process. This represents the initial population. Next, the population o Q-bit individuals is subject to a measurement procedure that results in an observed population, that is denoted by $P(t)$. Each element in this set corresponds to a binary chromosome. This is evaluated and the best solution in $P(t)$ are then stored. In that sense, the algorithm can be thought of following an elitist strategy.

```
procedure QEA
  t ← 0
  initialise Q(t)
  obtain initial P(t) by observing Q(t) states
  evaluate P(t)
  select best solution in P(t) and store in B(t)
  while (not-termination condition) do
      obtain P(t) by observing Q(t-1) states
      evaluate P(t)
      select best solution in P(t) and store in B(t)
      update Q(t) using quantum gates U(t)
      retain best solutions applying migration M(t)
      t ← t+1
  end while
end procedure
```

Figure 4.3 Pseudo-code for the QEA

The algorithm then enters a loop until a termination condition is satisfied. In each iteration (or generation), the set $Q(t)$ of Q-bits is updated. Recall that the Q-bits provide a probabilistic representation of states. Hence, the algorithm updates these probabilities so that the likelihood of observing states that are close to the best ones observed so far increases. This is achieved through the use of quantum rotation gates, which is explained in section 4.3.2.3.

The algorithm thus probabilistically moves the search towards the best solutions, which act as attractors. The choice of these attractors is also a critical component of the algorithm and is specified through the so-called migration strategies, detailed in section 4.3.2.4.

Figure 4.4 describes the block diagram of a QEA. It is also useful to represent the main steps of a conventional GA, as this will be used for comparison in the next chapter. In the canonical GA, the decision variables (genes) of a problem are coded in binary to form a chromosome. In the GA algorithm of Figure 4.5, five important operations can be identified: initialization, evaluation, selection, variation, which includes mutation and crossover operations, and replacement.

At generation $t = 0$, the initial population $P(t)$ is created at random, uniformly distributed over the entire search space. Next, $P(t)$ is evaluated according to a criteria (fitness function) and the promising solutions are selected. These candidate solutions are varied to generate new solutions sharing similarities with the original ones, but they are novel in some way. The new solutions replace the previous population and the iteration counter is updated (t ← t + 1). Operations from evaluation to replacement are iterated until the stop criterion is met.

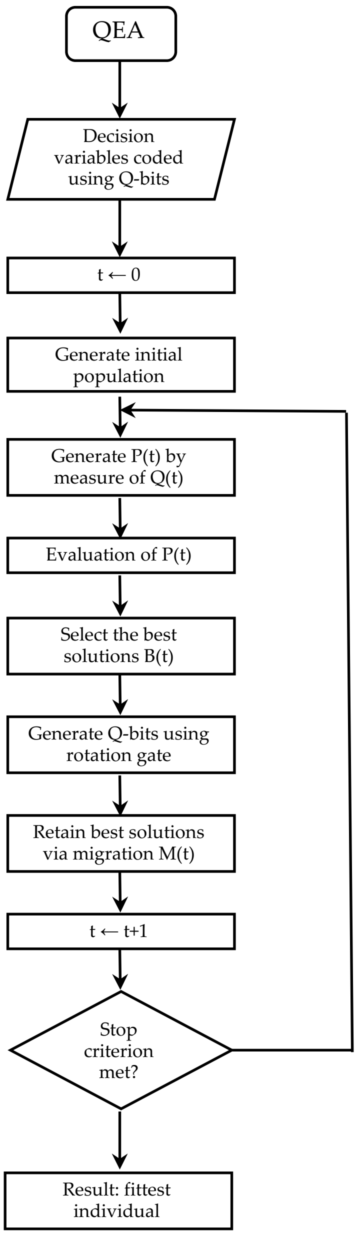

QEA
Decision variables coded using Q-bits
t ← 0
Generate initial population
Generate P(t) by measure of Q(t)
Evaluation of P(t)
Select the best solutions B(t)
Generate Q-bits using rotation gate
Retain best solutions via migration M(t)
t ← t+1
Stop criterion met?
Result: fittest individual

Figure 4.4 Block diagram for the QEA

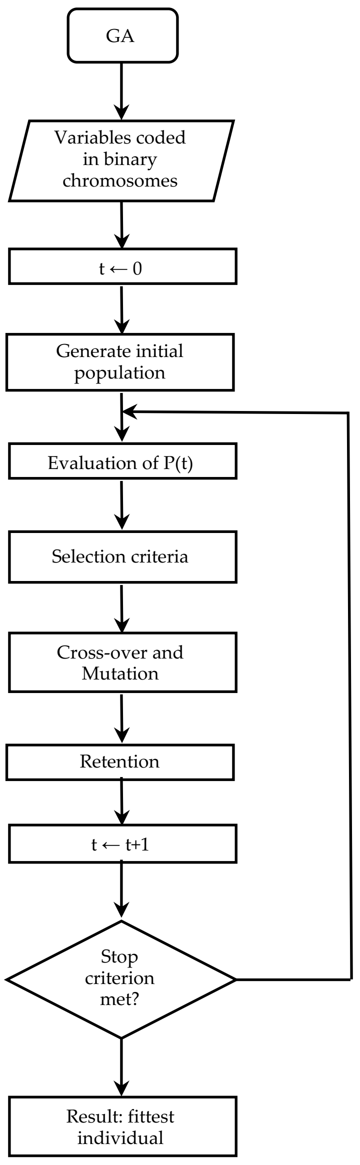


Figure 4.5 Block diagram for the canonical GA

4.3 The QEA for the TFPSN

The next subsections explain in some detail the several steps involved in the implementation of the QEA.

4.3.1 Initialisation

The QEA maintains a population of Q-bit individuals, $Q(t) = \{\boldsymbol{q}_1^t, \boldsymbol{q}_2^t, \dots, \boldsymbol{q}_n^t\}$ at generation t, where n is the size of the population and $\boldsymbol{q}_j^t$ is a Q-bit individual defined as:

$$\boldsymbol{q}_j^t = \begin{bmatrix} \boldsymbol{\alpha}_j^t \\ \boldsymbol{\beta}_j^t \end{bmatrix} = \begin{bmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \dots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \dots & \beta_{jm}^t \end{bmatrix},$$

where m is the size of the encoded solution, that is, the genotype. The basic steps of the initialisation process are summarised as follows:

- **Step 1 Initial Values:** One common approach to setting initial Q-bit values is to assign the same probability to each state. In this case, $\boldsymbol{\alpha}_j^0 = \boldsymbol{\beta}_j^0$ for $j = 1,\dots,n$ and each element is equal to $1/\sqrt{2}$. Accordingly, the initial Q-bit population is:

$$Q(0) = \{\boldsymbol{q}_1^0, \boldsymbol{q}_2^0, \dots, \boldsymbol{q}_n^0\},$$

where each initial Q-bit $\boldsymbol{q}_j^0 = \boldsymbol{q}^0$ for $j = 1,\dots,n$ and all Q-bit individuals are equal, hence:

$$\boldsymbol{q}^{\boldsymbol{0}} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & \dots & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & \dots & 1/\sqrt{2} \end{bmatrix}.$$

- **Step 2 Initial Observation:** This step makes binary solutions by observing the states of the initial population, $Q(0)$, of Q-bits. The set of observations are stored in:

$$P(0) = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \dots, \boldsymbol{x}_n^0\},$$

where each of the n elements $\boldsymbol{x}_j^0$ of $P(0)$ represents a binary solution (string) of length m, which is formed by choosing either 0 or 1 for each bit $i = 1,\dots,m$ using the probability, either $|\alpha_{ji}|^2$ or $|\beta_{ji}|^2$, of each Q-bit $\boldsymbol{q}_j^0$ for $j = 1,\dots,n$. The precise observation procedure will be described in detailed in Step 5.

- **Step 3 Initial Evaluation:** Each binary solution $\boldsymbol{x}_j^0$ is evaluated to give its level of fitness which is denoted as $F_j^0 = F(\boldsymbol{x}_j^0)$ for $j = 1,\ldots,n$. The set of function evaluations is stored at

$$F(0) = \{F_1^0, F_2^0, \ldots, F_n^0\}.$$

- **Step 4 Initial Selection / Retention:** The initial best solutions are then selected among the observed binary solutions, $P(0)$. These are stored into the set:

$$B(0) = \{\boldsymbol{b}_1^0, \boldsymbol{b}_2^0, \ldots, \boldsymbol{b}_n^0\}.$$

  As this is the first generation, all individuals of the population are selected meaning that $B(0) = P(0)$ and $\boldsymbol{b}_j^0 = \boldsymbol{x}_j^0$ for $j = 1,\ldots,n$. Additionally, the overall best solution in $B(t)$, denoted by $\boldsymbol{b}$, that is,

$$\boldsymbol{b} = \arg\max\{F(\boldsymbol{b}_1^0), F(\boldsymbol{b}_2^0), \ldots, F(\boldsymbol{b}_n^0)\}$$

  is also stored (retained).

It is important to highlight the role of the selection / retention process. These elements will play the role of attractor for the algorithm. The choice of the attractor will influence the speed of convergence of the algorithm.

There are several options for the choice of the attractor. For instance, one could consider a global attractor that is used by all the individuals in the population. In this case, the element candidate solution $\boldsymbol{b}$ would act as the global attractor. Another possibility is to assign to each individual its own attractor. In this case, each individual $j = 1,\ldots,n$ would have a specific attractor, in this case, this would be $\boldsymbol{b}_j^0$.

There are other possibilities that could also be considered. For instance, one could divide the population into smaller groups that share the same attractor, or there could design situations where attractors are exchanged amongst groups. This will be discussed in section 4.3.2.4 on migration strategies.

4.3.2 Main Loop

Once initial values and parameters are set, then the algorithm enters into a loop over generations, indexed by t. The loop keeps going until a termination condition is met. Each step of the loop can be associated to an evolutionary cycle.

In each generation, the process involves observing the current population of Q-bit individuals, evaluating the observed solutions, selecting and retaining the best ones and updating the population of Q-bit individuals for its use at the next iteration.

4.3.2.1 Evaluation

In this step, a population of candidate solutions is obtained by "measuring" the Q-bit individuals. This is expressed formally as:

- **Step 5 Observation:** At each iteration in the loop, a set $P(t)$ of candidate binary solutions is formed by making an observation of the Q-bit individuals computed in the previous iteration, $Q(t-1)$.

One of the distinctive elements of quantum computing is the collapse or realisation of states through the process of observation. In the canonical QEA, this is achieved through the following process:

a) For every bit x_{ji}^{t} with $i = 1,\ldots,m$ in the binary string $\boldsymbol{x}_{j}^{t}$ with $j = 1,\ldots.,n$, a random number, denoted as r, is generated from the range $[0,1)$;

b) If $r < |\beta_{ji}|^2$ then the bit of the binary string is set to 1, otherwise it is set to 0.

Thus, a binary string of length m is formed from the Q-bit individual, which represents a potential solution observed from the j-th Q-bit individual.

- **Step 6 Evaluation:** Each observed string in $P(t)$ is then evaluated and the corresponding fitness value obtained. This is denoted as $F_j^t = F(\boldsymbol{x}_j^t)$ for $j = 1,\ldots,n$. The set of function evaluations is stored at

$$F(t) = \{F_1^t, F_2^t, \ldots, F_n^t\}.$$

4.3.2.2 Selection

Once the elements of the population have been observed and evaluated, a selection mechanism takes place in order to identify the fittest individuals:

- **Step 7 Selection:** Each candidate solution $\boldsymbol{x}_j^t$ in $P(t)$, with $j = 1,\ldots,n$, and the corresponding previous best solutions $\boldsymbol{b}_j^{t-1}$ stored in $B(t-1)$ are compared and the better ones are saved into $B(t)$. The overall best solution in $B(t)$ is also stored as $\mathbf{b}$.

This selection mechanism resembles an elitist procedure in conventional GAs. The best solution is chosen as an attractor.

4.3.2.3 Variation

A key element of the QEA is the process that updates the population of Q-bit individuals. This is very important as it will determine the search carried out by the algorithm and thus the process for getting closer to fitter solutions. Indeed, as noted by Zhang et al. (2011), compared with binary, numeric and symbolic representations, the Q-bit representation can achieve a linear superposition of states given its probabilistic approach.

Using a Q-gate as a variation operator, instead of crossover, recombination and mutation operators, the basic QEA can find the optimal or close-to-optimal solutions with a small number of individuals, even with a single individual via multiple observations of the system.

- **Step 8 Update:** The current population of Q-bit individuals is updated by applying Q-gates to obtain a new set $Q(t)$. A Q-gate is a variation operator of quantum evolutionary algorithms. By this operation, the probabilities of the updated Q-bit must satisfy the normalisation condition.

In the canonical QEA, the update is implemented through a rotation gate.[12] To understand this concept, it must be noted that the Q-bits can be represented using a polar coordinate system. Two elements are needed for this purpose: an angular coordinate, denoted as θ, and a radial coordinate which is equal to one in this case. Figure 4.6 shows the representation of a Q-bit individual.

[12] The rotation operator can be identified with the quantum principle of interference. See Lahoz-Beltra (2016) for an exposition on this topic.

The idea of the rotation update is to change the direction of the Q-bit individual, that is, the angle θ, so that the chance of observing a state eventually leading to a fitter solution increases. Hence, the key design problem in QEAs is the choice of the rotation angle. It must be noted that rotation value plays the role of an "evolution rate". This parameter also plays a significant role in determining the convergence of the algorithm. Consequently, when setting the rotation angle, one should avoid assigning too high or too low values.

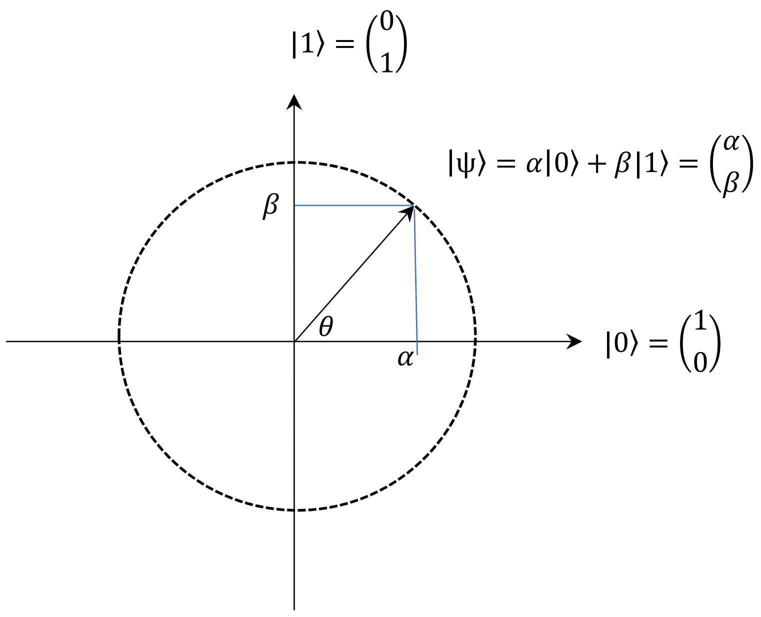

$|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$
β
θ
α
$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$

Figure 4.6 Polar plot for a Q-bit individual

Figure 4.7 illustrates the mechanics of the vector rotation. Given an initial state characterised by the vector (α_0, β_0) and the associated angle θ_0 we have that: $\alpha_0 = \cos\theta_0$ and $\beta_0 = \sin\theta_0$. This vector is then rotated by a magnitude equal to $\Delta\theta$ leading to the new vector (α_1, β_1) with an associated angle $\theta_1 = \theta_0 + \Delta\theta$.

Each component of the new, rotated vector can be expressed as $\alpha_1 = \cos\theta_1$ and $\beta_1 = \sin\theta_1$ or equivalently as $\alpha_1 = \cos(\theta_0 + \Delta\theta)$ and $\beta_1 = \sin(\theta_0 + \Delta\theta)$. Taking into consideration the following trigonometric identities:

$$\cos(\theta_0 + \Delta\theta) = \cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta)$$

and

$$\sin(\theta_0 + \Delta\theta) = \sin(\theta_0)\cos(\Delta\theta) + \cos(\theta_0)\sin(\Delta\theta).$$

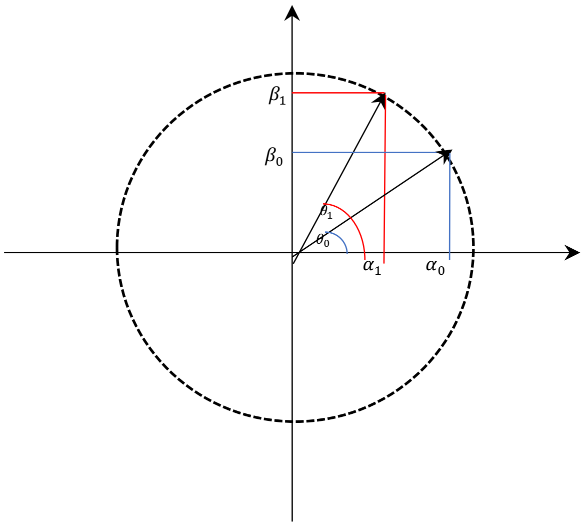


Figure 4.7 Polar plot of vector rotations

Hence, carrying out the substitutions, the new vector coordinates can be expressed as:

$$\alpha_1 = \alpha_0 \cos(\Delta\theta) - \beta_0 \sin(\Delta\theta)$$

and

$$\beta_1 = \beta_0 \cos(\Delta\theta) + \alpha_0 \sin(\Delta\theta).$$

More generally, for each element of each Q-bit $\boldsymbol{q}_{ij}^t = \left[\alpha_{ij}^t, \beta_{ij}^t\right]^T$ with $j = 1,\ldots,n$ and $i = 1,\ldots,m$, the following update procedure is used to obtain the Q-bit $\boldsymbol{q}_{ij}^{t+1} = \left[\alpha_{ij}^{t+1}, \beta_{ij}^{t+1}\right]^T$:

$$\begin{bmatrix} \alpha_{ij}^{t+1} \\ \beta_{ij}^{t+1} \end{bmatrix} = G_{ij}^t(\theta) \begin{bmatrix} \alpha_{ij}^t \\ \beta_{ij}^t \end{bmatrix},$$

where $G_{ij}^t(\theta)$ is a Q-gate defined by the rotation matrix

$$G_{ij}^t(\theta) = \begin{bmatrix} \cos\theta_{ij}^t & -\sin\theta_{ij}^t \\ \sin\theta_{ij}^t & \cos\theta_{ij}^t \end{bmatrix},$$

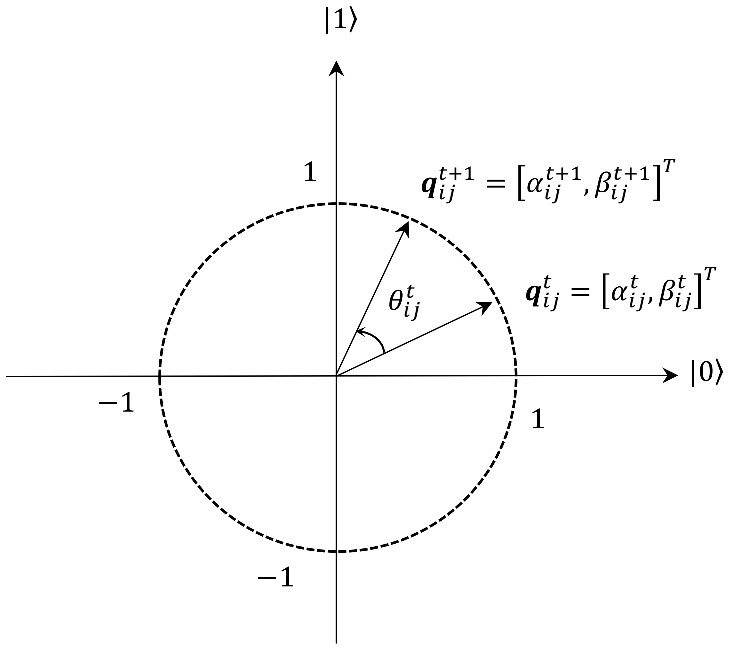
and θ_{ij}^t is an adjustable rotation angle.

The objective of the rotation is to obtain a higher probability of generating solution strings which are similar to the best solution in subsequent iterations. There are two elements to take into account in this process: the direction and the size of the angle rotation. Formally, the rotation angle is expressed as follows:

$$\theta_{ij}^{t} = s\left(\alpha_{ij}^{t}, \beta_{ij}^{t}\right)\Delta\theta_{ij}^{t}$$

where $s\left(\alpha_{ij}^{t}, \beta_{ij}^{t}\right)$ denotes the direction of the rotation, which depends on the location (quadrant) of the Q-bit in the polar coordinate system; and, the size of the rotation, which is denoted by $\Delta\theta_{ij}^{t}$.

Figure 4.8 illustrates a polar plot of the rotation gate for a Q-bit individual. The idea of the Q-Gate is to update the elements of a Q-bit string so that they move towards the best solution, which acts as an attractor in this process. The specific choice of attractor will depend on the migration strategies selected as described below in sub-section 4.4.5.



$|1\rangle$
1
$\boldsymbol{q}_{ij}^{t+1} = \left[\alpha_{ij}^{t+1}, \beta_{ij}^{t+1}\right]^{T}$
θ_{ij}^{t}
$\boldsymbol{q}_{ij}^{t} = \left[\alpha_{ij}^{t}, \beta_{ij}^{t}\right]^{T}$
$|0\rangle$
-1
1
-1

Figure 4.8 Polar plot of the rotation gate for a Q-bit individual

In order to help understand the process, let us consider a maximisation problem with a candidate solution vector $\boldsymbol{x}_{j}^{t}$ and the corresponding attractor $\boldsymbol{b}_{j}^{t}$, the update process involves analysing each element of $\boldsymbol{x}_{j}^{t}$. For instance, the case where $x_{ji}^{t} = 1$ and $b_{ji}^{t} = 0$, if the objective function value of the best solution is better than the objection function value of the current solution, that is, $F\left(\boldsymbol{b}_{j}^{t}\right) \geq F\left(\boldsymbol{x}_{j}^{t}\right)$, then:

(i) If the Q-bit is located in the first or the third quadrant in Figure 4.6, that is, both α_{ij}^{t} and β_{ij}^{t} have the same sign either positive or negative, then the value of the rotation angle should be set to a negative value (move towards the x-axis) so that the probability of $\boldsymbol{q}_{ij}^{t}$ to collapse to the state $|0\rangle$ is increased.

(ii) If the Q-bit is located instead in the second or the fourth quadrant in Figure 4.6, the angle should be increased, that is, moved counter clock-wise towards the x-axis so that the probability of $\boldsymbol{q}_{ij}^{t}$ to collapse to the state $|0\rangle$ is increased.

Next, consider the case when $x_{ji}^{t} = 1$ and $b_{ji}^{t}10$ and as before $F(\boldsymbol{b}_{j}^{t})$ is better than $F(\boldsymbol{x}_{j}^{t})$, then:

(i) If the Q-bit is located in the first or the third quadrant in Figure 4.6, then the angle should be increased, that is, rotated towards the y-axis, so that the probability of $\boldsymbol{q}_{ij}^{t}$ to collapse to the state $|1\rangle$ is increased.

(ii) If the Q-bit is located in the second or the fourth quadrant in Figure 4.6, then the angle should be reduced, that is, moved counter clock-wise in order to increase the probability of $\boldsymbol{q}_{ij}^{t}$ to collapse to the state $|1\rangle$.

Table 4.1 presents a general look-up table for the QEA with the information on the direction of movement of the rotation angle for all the possible cases. For each of these cases, it is necessary to determine the size of the rotation. This is an important aspect of QEA that requires particular attention as described, for instance, in Mani et al. (2017).

It must be pointed out that the specific magnitude of the rotation angle will depend on the problem considered. The magnitude of the rotation angle also influences the speed of convergence of the algorithm. Depending on the value considered for this parameter, the solutions may diverge or converge prematurely to a local optimum.

xi	bi	f(b) better than f(x)	alpha	beta	Quadrant	Direction	Size	Remarks
0	0	TRUE	+	+	1	-	θ1	State vector should be rotated slightly towards |0⟩ as bi=0, so in next iteration also, ith Q-bit, qi should have an improved probability of collapsing |0⟩.
			-	+	2	+		
			-	-	3	-		
			+	-	4	+		
0	0	FALSE	+	+	1	-	θ2	State vector should be rotated slightly towards |0⟩ as with xi=0 it has found a better solution so in next iteration also, i-th Q-bit, qi should have an improved probability of collapsing |0⟩.
			-	+	2	+		
			-	-	3	-		
			+	-	4	+		
0	1	TRUE	+	+	1	+	θ3	State vector should be rotated adequately towards |1⟩ as bi=1 so in next iteration, ith Q-bit, qi should have an improved probability of collapsing |1⟩.
			-	+	2	-		
			-	-	3	+		
			+	-	4	-		
0	1	FALSE	+	+	1	-	θ4	State vector should be rotated slightly towards |0⟩ as with xi=0, it has found a better solution so in the next iteration also, ith Q-bit, qi should have an improved probability of collapsing |0⟩.
			-	+	2	+		
			-	-	3	-		
			+	-	4	+		
1	0	TRUE	+	+	1	-	θ5	State vector should be rotated adequately towards |0⟩ as bi=0 so in next iteration ith Q-bit, qi should have an improved probability of collapsing |0⟩.
			-	+	2	+		
			-	-	3	-		
			+	-	4	+		
1	0	FALSE	+	+	1	+	θ6	State vector should be rotated slightly towards |1⟩ as with xi=1, it has found a better solution so in next iteration also ith Q-bit, qi should have an improved probability of collapsing |1⟩.
			-	+	2	-		
			-	-	3	+		
			+	-	4	-		
1	1	TRUE	+	+	1	+	θ7	State vector should be rotated slightly towards |1⟩ as bi=1 so in next iteration also this ith Q-bit, qi should have an improved probability of collapsing |1⟩.
			-	+	2	-		
			-	-	3	+		
			+	-	4	-		
1	1	FALSE	+	+	1	+	θ8	State vector should be rotated slightly towards |1⟩ as with xi=1 it has found a better solution so in the next iteration also ith Q-bit, qi should have an improved probability of collapsing |1⟩.
			-	+	2	-		
			-	-	3	+		
			+	-	4	-		

Table 4.1 General Look-up table for the QEA

4.3.2.4 Retention

QEAs use the current best solution to control different searching directions. The best solutions are kept so that in next iterations the likelihood of moving towards fitter solutions is increased. These retained best solutions act as an attractor.

Recall that each candidate solution vector $\boldsymbol{x}_j$ has associated a corresponding attractor $\boldsymbol{b}_j$. The question thus is how to determine these focal points. There are different approaches to determine the attractors. One approach assigns a common attractor to all the individuals in the population. This is called global migration, as all the elements of the set $B(t)$ "migrate" to this point of reference.

Another approach consists of assigning to each individual in the population its own attractor. In this context, every solution $\boldsymbol{x}_j$ is compared, by means of objective function evaluations, to the corresponding attractor. If the resulting value is better, then the attractor is replaced by the current candidate solution. The process is repeated in each iteration until the termination condition is met. This approach is known as individual migration.

The two approaches discussed above will be considered in Chapter 5 when performing the evaluation of the QEA in the context of the TFPSN. Hence, the migration step could be formally stated as follows:

- **Step 9 Migration:** This step includes individual and global migrations, where a migration is defined as the process of copying either $\boldsymbol{b}_j^t$ in $B(t)$ or $\mathbf{b}$ to $B(t)$. A global migration is realized by substituting $\mathbf{b}$ for all the solutions in $B(t)$, and an individual migration is realized for each solution in $B(t)$, that is, by substituting the better one between the current solution and the attractor $\boldsymbol{b}_j^t$.

It must be pointed out that there are other migration strategies. For instance, one method consists of forming sub-groups within the population. Accordingly, each of these groups would have their own attractor. In each iteration, the previous group-wide attractor is compared to each element of the group. If any of the candidate solutions has a better fit than the current attractor, this solution would become the new attractor for the group.

Other strategies would involve situations where attractors are exchanged amongst groups. These type of approaches are known as local migrations where only a small amount of information needs to be exchanged between multiple subpopulations.[13]

Alternative migration strategies can be illustrated following *Nakayama (2006)*. Specifically, Figure 4.9 shows the local migration whereby each of the two sub-groups that divide the population have their own attractor and this is the one used only for the individuals in that sub-group; Figure 4.10 illustrates the case of global migration where there is a unique attractor that is used for all individuals in the population.

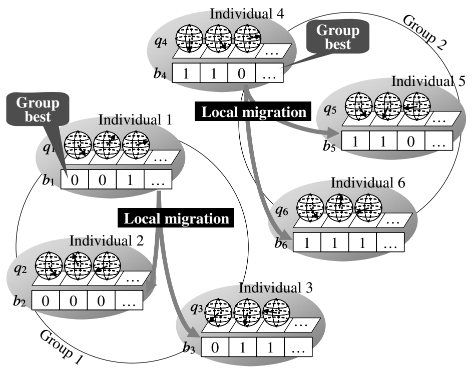
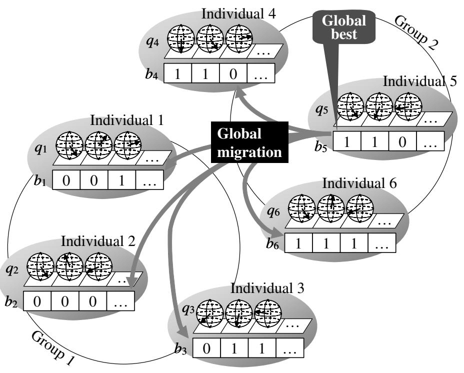


Figure 4.9 Illustration of a local migration strategy



[13] Because of this property, QEAs are suitable for parallel implementation and have the potential to greatly reduce the communication and synchronization costs.

Figure 4.10 Illustration of a global migration strategy

Finally, it must be pointed out that the binary solutions in $P(t)$ are discarded at the end of the iteration. This is so because the candidate solutions for the next generation, $P(t+1)$, will be produced by observing the updated Q-bit.

4.3.3 Termination

Until the termination condition is satisfied, the algorithm is iteratively running in the while loop. Whereas standard evolutionary algorithms generally use the maximal number of generations as a termination condition, QEAs could employ a Q-bit convergence termination criterion due to its probability-based representation of the individuals. For instance, the following condition can be applied:

- **Step 10 – Termination:** The algorithm finalises the search of solutions whenever the following condition is met:

  $$\mathrm{Prob}(\mathbf{b}) > \gamma_0,$$

  where

  $$\mathrm{Prob}(\mathbf{b}) = \frac{1}{n}\sum_{j=1}^{n}\left(\prod_{i=1}^{m} p_{ji}\right)$$

  with

  $$p_{ji} = \begin{cases} |\alpha_{ji}|^2 & \text{if} \quad b_i = 0 \\ |\beta_{ji}|^2 & \text{if} \quad b_i = 1 \end{cases}$$

  and b_i is the i-th bit of the best solution $\mathbf{b}$.

The probability $\mathrm{Prob}(\mathbf{b})$ represents the average convergence of all Q-bit individuals to the best solution. The parameter γ_0 will determine the time to achieve convergence. It is usually set to a value above 0.9. The termination condition gives a clear meaning to how much closely Q-bit individuals converge to either 0 or 1.

Having reached this point and taking into consideration the exposure of the algorithm presented in the preceding sections, the general process of the QEA can be summarised as in Figure 4.11.

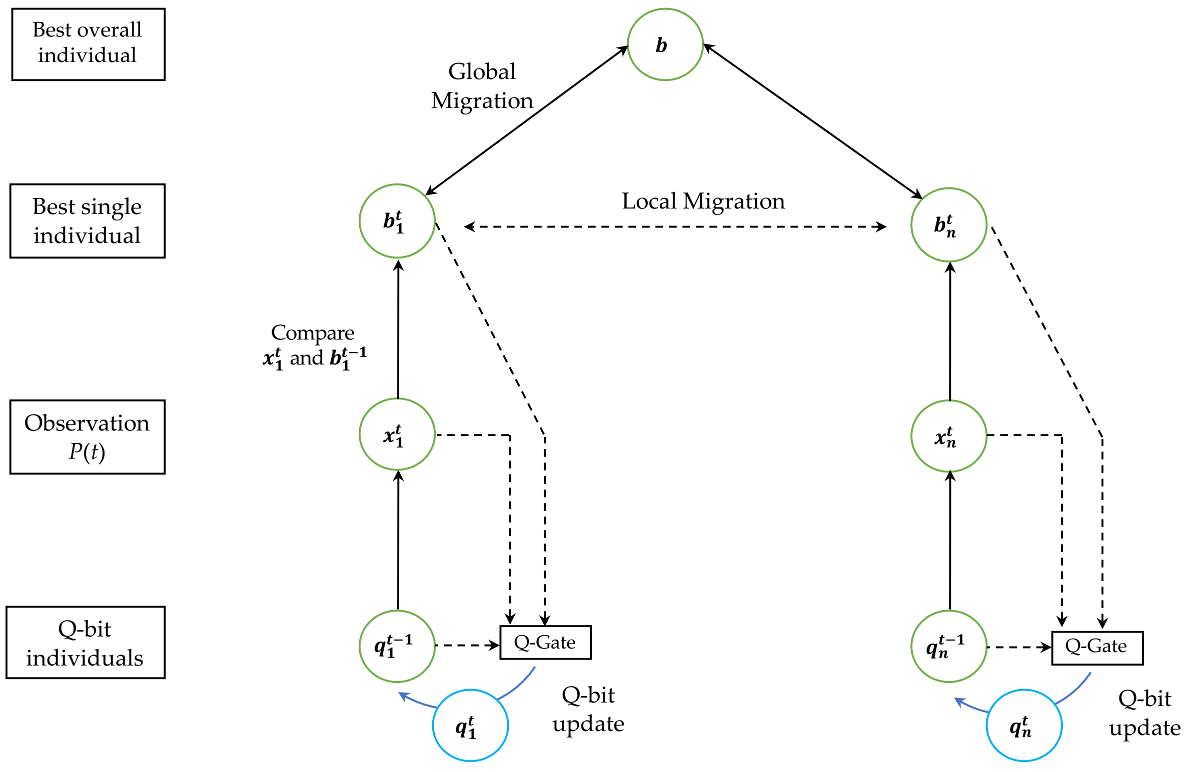


Figure 4.11 Overall structure of QEAs as described in Han and Kim (2002)

4.4 Formulation of the TFPSN in the context of the QEA

This section describes how the TFPSN can be adapted for its solution by the QEA described above. There are two main elements in this regard. The first one is the representation of the problem, while the second component is related to the design of the fitness function.

4.4.1 Representation of the Problem

In order to guide the exposition, the examples illustrated in Figure 2.1 will be taken as reference. In that example, a network was formed by six experts $V = \{v_1, \ldots, v_6\}$. Hence, each candidate solution in this example would be represented by a vector of size 6, where each component would be either 1 or 0 depending on whether an individual is chosen or not to form part of the team:

x_1	x_2	x_3	x_4	x_5	x_6

where x_i is a binary variable. In this particular example, two possible teams could be formed for a given task *T* that required five skills $\{s_1, s_2, s_3, s_4, s_5\}$. These teams were $X_1 = \{v_1, v_2, v_3, v_4\}$ and $X_2 = \{v_3, v_4, v_5, v_6\}$. Hence, the solutions could be represented as follows:

1	1	1	1	0	0

and

0	0	1	1	1	1

Recall that the aim of the TFPSN is to find a team of experts denoted by $X \subseteq V$, that collectively cover all the required skills with the least collaboration cost. It could be the case that one particular person has more than one of the required skills. Without loss of generality, we will assume that each individual has only one skill. In the case of a person having, for instance, two skills, we could consider this as two "separate" individuals.[14]

4.4.2 Definition of the Fitness Function

Once the possible solutions have been represented, the next step for the specification of the TFPSN is the calculation of the fitness function. In this case, the function represents the communication costs amongst the members of the team. The collaboration cost is represented through the closeness of the experts in a social network $G(V,E)$. The collaboration cost of two adjacent experts, v_i and v_j, is given by the cheapest path between them, denoted as $sp(v_i,v_j)$.

As described in Section 2.2, there are several specifications widely used in the literature. In the present application, the following two will be considered:

$$Cc\text{-}D(X) = \max_{v_i, v_j \in X} sp(v_i, v_j)$$

[14] In the case that these skills can be used simultaneously in the given problem, we would assume zero collaboration cost between the two-person representation of this particular expert. However, in those cases where the expert can only provide one single skill at a time, then a very large collaboration cost would be assumed.

and

$$Cc\text{-}SD(X) = \sum_{v_i, v_j \in X} sp(v_i, v_j).$$

The first one is the diameter distance, which is defined as the maximum collaboration cost among the cost of the cheapest paths between any two experts in the team X. The second represents the sum of distances that measures the sum of all the cheapest paths between any two experts in team X.

The next chapter describes the results corresponding to a numerical evaluation of this problem.

Chapter 5 Evaluation

This chapter presents the results from a numerical evaluation of the algorithm described in the previous chapter. A key issue in designing a good evolutionary algorithm is the management of the exploration versus exploitation balance. A good algorithm must use, or exploit, already discovered fit solution encodings, while not neglecting to continue to explore new regions of the search space which may contain even better solution encodings. This will be one of the focus of the evaluation.

One possible way of measuring the degree of diversity is through the analysis of convergence of the population. Morrison and De Jong (2002) proposed the use of pair-wise distance metrics to this end. One metric that is commonly used is the so-called Hamming distance which, essentially, involves counting up which set of corresponding digits or places are different, and which are the same. It measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. Given two vectors of dimension n, x_1 and x_2, the Hamming distance between these two vectors is defined as follows:

$$\text{Hamming}(x_1, x_2) = \frac{1}{n}\sum_{i=1}^{n} d_i$$

where

$$d_i = \begin{cases} 0 \text{ if } x_{1i} = x_{2i} \\ 1 \text{ if } x_{1i} \neq x_{2i} \end{cases}$$

In order to carry out the numerical analysis, a set of synthetic data is generated first. Several instances of the TFPSN are considered whose complexity is defined by the number of experts as well as the number of required skills. Once the pool of experts is generated, an analysis is performed for a baseline GA and a QEA.[15]

[15] Numerical results were obtained using Matlab R2015b running on a 2 Ghz. Intel Core 2 Duo MacBook computer using OS X 10.6 with 8 GB of RAM.

5.1 Experimental Setup

In order to carry out the numerical experiments, several pools of experts were constructed. It was assumed that each expert had one skill only and that completing a given task requires having, at least, one expert for each of the required skills. It must be pointed out that an expert that has none of the required skills will not be considered in the pool of potential candidates. In this regard, one could think about a situation whereby two experts that have never collaborated before use a common past collaborator as a "connector". However, if this third party has none of the required skills for the task at hand, he or she will not be considered in the pool of potential team members. Also, isolated experts, that is, experts that have collaborated with none of the other experts before will not be considered as suitable candidates. This is a common assumption in the literature as presented, for instance, in Lappas et al. (2009).[16]

Next, for each pool of experts that satisfies the required set of skills, a collaboration network (a graph) is randomly generated. This is expressed as an adjacency matrix, where each entry indicates the direct collaboration cost between any two experts in the pool. The costs are obtained through pseudorandom values drawn from the standard uniform distribution over the open interval (0,1). Figure 5.1 illustrates an instance of a pool of experts generated using the procedure described above. In this case, all experts are connected amongst themselves and the values of each arch represent the corresponding collaboration cost.

In order to make the problem more realistic, it is assumed that not all experts are connected, that is, they have never worked together or they are incompatible. This is achieved by considering a threshold for the collaboration cost. If this cost turns out to be above a given value, then it is assumed that the experts are totally disconnected. Figure 5.2 illustrates this case, where *expert 3* is only able to collaborate with *expert 2* hence the latter can only be in a team where the former also forms part of.

[16] This might seem as a restrictive assumption, however, the aim of the TFPSN is to highlight the potential benefits of working in collaboration with other experts.

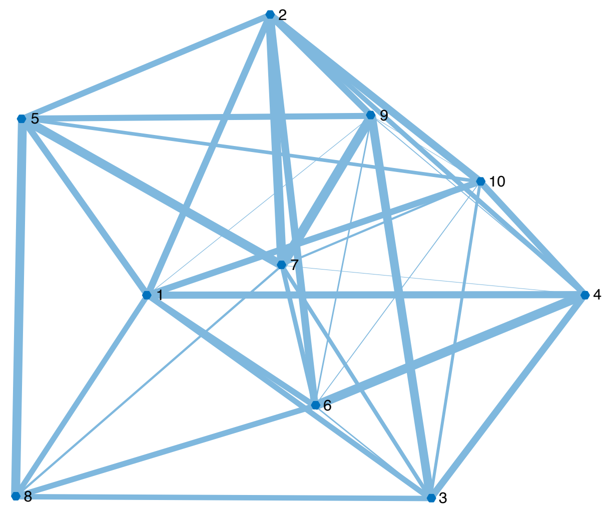


Figure 5.1 Randomly generated pool of 10 experts

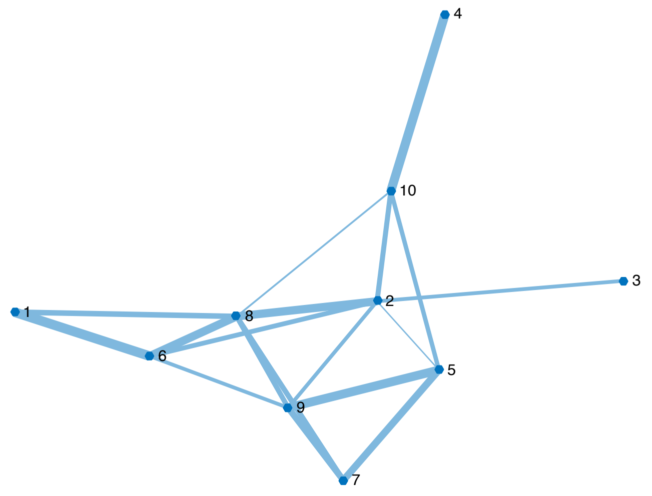


Figure 5.2 Sparse graph with 10 experts and a cost threshold of 0.6

The difficulty of the problem grows exponentially with the number of experts. As illustrated in Figure 5.3, the resulting graph can be very dense and complex.

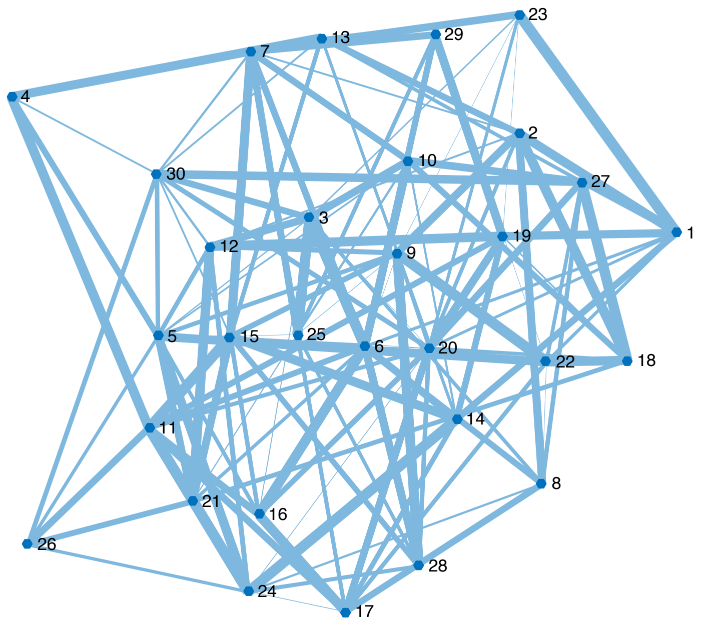


Figure 5.3 Graph of a pool consisting of 30 experts

Another factor that determines the difficulty of the problem refers to the number of skills required for solving a given task. It must be reiterated that each expert has one skill only, but in the pool of experts, there could be several experts with the same skill.

For the numerical evaluation carried out in this work, several configurations of the TFPSN are considered. These versions of the problem differ on the number of available experts, as well as on the number of required skills. Table 5.1 provides the specific instances considered.

Problem	Number of Available Experts	Number of Required Skills
1	50	10
2		5
3	100	20
4		5
5	300	30
6		5

Table 5-1 Configurations of the TFPSN considered in the analysis

5.2 Evaluation of a Genetic Algorithm

In order to establish a reference for the evaluation of the QEA in the context of the TFPSN, a standard genetic algorithm is considered first. There are three key elements that define the algorithm: parent selection based on the tournament method; variation generation through single-point crossover and bit-flip mutation; retention based on elitism where the best individuals are kept in the population of candidate solutions and all the rest are replaced. Table 5.2 lists the key parameters for this algorithm whose values are discussed below.

Parent Selection		Method	Tournament
		Size (n_{ts})	10% and 30%
Variation	**Crossover**	Operator	Single point
		Rate (p_c)	0.6 and 0.9
	Mutation	Operator	Uniform Bit-flip
		Rate	0.01 and 0.1
Retention		Method	Elitism
		Size	Best individual

Table 5-2 Parameter values for the baseline genetic algorithm

5.2.1 Configuration of the Parameter Values

Regarding parent selection, the parameter to choose is the tournament size, denoted as n_{ts}. This parameter represents a group of n_{ts} individuals randomly chosen from the population, where $n_{ts} < n$, and n is the total number of individuals in the population. The fitness of the selected n_{ts} individuals is compared and the best individual from this group is selected and returned by the operator. Provided that the tournament size is not too large, tournament selection prevents the best individual from dominating, thus having a lower selection pressure. On the other hand, if n_{ts} is too small, the chances that bad individuals are selected increase. There are several methods to choose the tournament size. One common approach is to assume a fixed tournament size. If $n_{ts} = n$ then the best individual will always be selected. On the other hand, if $n_{ts} = 1$, random selection is obtained. Perhaps the most common approach is to choose a binary tournament where $n_{ts} = 2$. However, there are other alternative methods that try to improve the performance of the GA by specifying a more sophisticated mechanism for determining the tournament size. For instance, Vajda et al. (2008) review tournament size methods such as Time-varying deterministic and Self-adaptive.

In the present work, we will analyse the effect of different tournament sizes on the results of the TFPSN. The tournament size will be expressed as a percentage of the total population, which remains fixed in the GA considered here, instead of an absolute value. Specifically, the values considered for n_{ts} will range between 10% and 40%. It must be pointed out that, in the experiments, the population was equal to 50 individuals and thus the tournament size was formed by integer numbers.

Crossover allows for the inheritance of groups of "good genes" or building blocks by the offspring of parents, thereby encouraging more intensive search around already discovered good solutions. In terms of single-point crossover, the parameter to be chosen, denoted as p_c is set at the start of the GA. The crossover mechanism works as follows: For each pair of selected parents, a random number is generated from the uniform distribution in the open interval (0,1), If this value is lower than p_c, crossover is applied to generate two new children; otherwise crossover is bypassed and the two children are clones of their parents. Crossover rates are typically selected from the range $p_c \in (0.6, 0.9)$.

The mutation operator plays also a relevant role as it can potentially uncover useful novelty.[17] The parameter to choose is the rate or probability of mutation, denoted as p_m. If a very high rate is applied, the selection and crossover operators can be overpowered and the GA will

[17] In contrast, crossover, if applied as a sole method of generating diversity, ceases to generate novelty once all members of the population converge to the same genotype.

effectively resemble a random search process. Conversely, if a high selection pressure is used, a higher mutation rate will be required in order to prevent premature convergence of the population. In setting an appropriate rate of mutation, the aim is to select a rate which helps generate useful novelty but which does not rapidly destroy good solutions before they can be exploited through selection and crossover. For each gene, a random number is generated from the uniform distribution in the interval (0,1). The binary gene is changed if this random number is less than p_m. In general, the values chosen for p_m tend to be low, in the range that allows from a gene per population to a gene per individual to be mutated.

5.2.2 Results

Based on the parameterisation discussed above, a series of simulations were performed. The results are summarised in Table 5.3 for the specific parameter configuration where the tournament size $n_{ts} = 25\%$, the cross over rate $p_c = 0.75$, and, the mutation rate $p_m = 0.05$. For each version of the problem, 20 simulations were carried out, and statistical measures over the best fitness values at the final generation were taken. The size of the population was set to 50 individuals and the maximum number of generations to 1500.

TFPSN		Best Collaboration Cost (sum of distances)				Hamming Distance	Team Size / Skills
Problem	Dimension	Min	Max	Median	Std. Dev.		
1	50-10	19.16	22.87	20.36	1.12	0.27	10/10
2	50-5	0.79	1.93	0.99	0.33	0.2	5/5
3	100-20	117.57	137.55	127.87	5.42	0.3	20 / 20
4	100-5	1.81	4.43	2.82	0.58	0.18	5/5
5	300-30	884.58	1211.11	1087.1	85.76	0.27	52/30
6	300-5	255.59	449.09	376.43	64.27	0.24	32/5

Table 5-3 Results for the baseline genetic algorithm

The results show that the GA achieves a relative good performance for small-size problems. For instance, Problem 1 consists of 50 experts with 10 skills. In this case, the algorithm finds a team with the desired number of skills and there are no duplicated roles, that is, each member of the team has different skills. The collaboration cost is low with a median value over the simulations of 20.36. The corresponding standard deviation is also low. Similarly, the median Hamming distance, which is considered to be a measure of diversity, turns out to be low. A similar pattern is observed for Problems 2 to 4.

Figure 5.4 illustrates the progress curve for Problem 1. The GA converges relatively quickly to an optimal solution in terms of minimum collaboration cost. Figure 5.5 shows the evolution of the Hamming distance, which measures the diversity in the population. The results indicate after an initial and significant drop, the degree of diversity is relatively stable. However, it can be observed that there are some fluctuations that can be interpreted as relative explorations of the solution space.

Certainly, as the number of experts and/or the number of required skills increase, the collaboration cost is higher and the algorithms show more variability over simulations. The standard deviation increases, notably for Problem 3 which tries to form teams of 20 experts. However, in all three cases, the GA finds a team with the minimum necessary number of members.

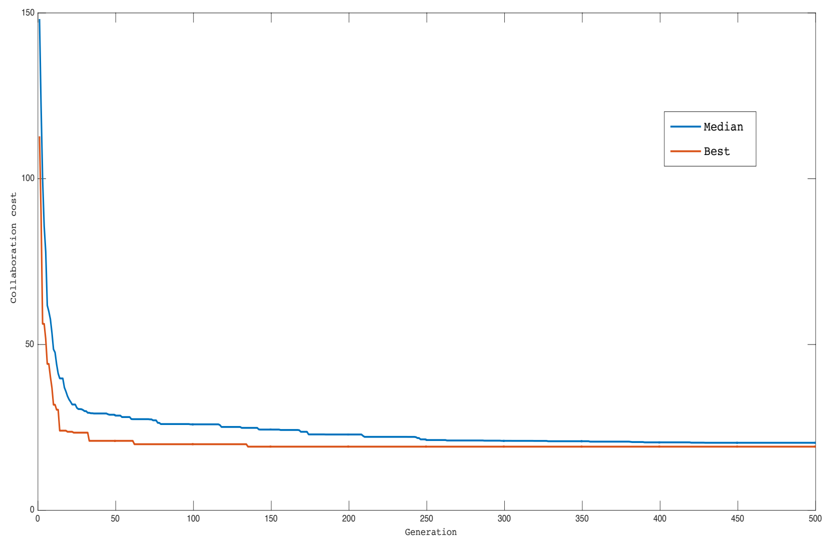


Figure 5.4 Problem 1: Progress Curve

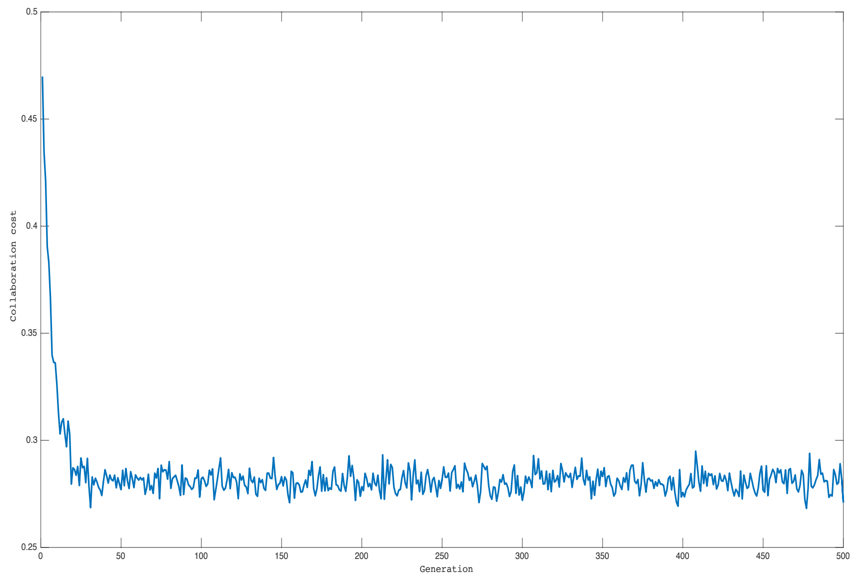


Figure 5.5 Problem 1: Hamming distance

As the complexity of the problem increases, the GA algorithm has difficulty in forming good teams. For instance, for Problem 6, where there is a pool of 300 experts to form teams with 5 skills, the algorithm achieves at best a team with 32 members. Hence, there is a duplicity of skills which obviously is not desirable from a cost minimisation perspective. The results also show a large variability over simulations as reflected in the relatively high standard deviation.

A similar result is observed for Problem 5, where the number of required skills is set to 30. Figure 5.6. shows the progress curve representing the evolution of the fitness value (in this case, the collaboration cost) for successive generations. In this case, the collaboration costs falls significantly in the first few generations, but remains relatively flat thereafter. The Hamming distance, shown in Figure 5.7, displays a similar pattern. However, in comparison with the results of Problem 1, the observed reduction in the amount of diversity is less pronounced in the initial phases (generations) of search process.

The results discussed above correspond to a baseline parameterisation. However, given the complexity of the problem at hand, it might be necessary to consider alternative parameter values. Choices for the selection strategy, the design of mutation and recombination operators, and the replacement strategy, determine the balance between exploration and exploitation. Selection and crossover tend to promote exploitation of already-discovered information, whereas mutation tends to promote exploration.

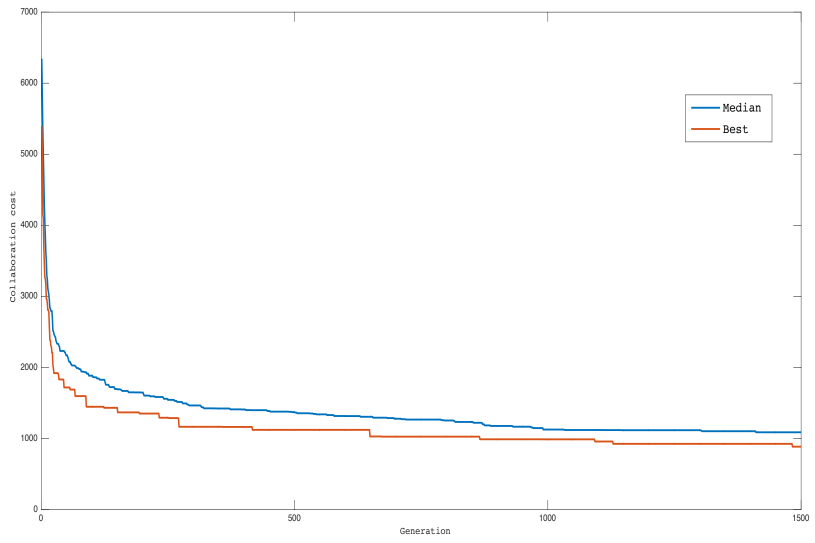


Figure 5.6 Problem 5: Progress curve

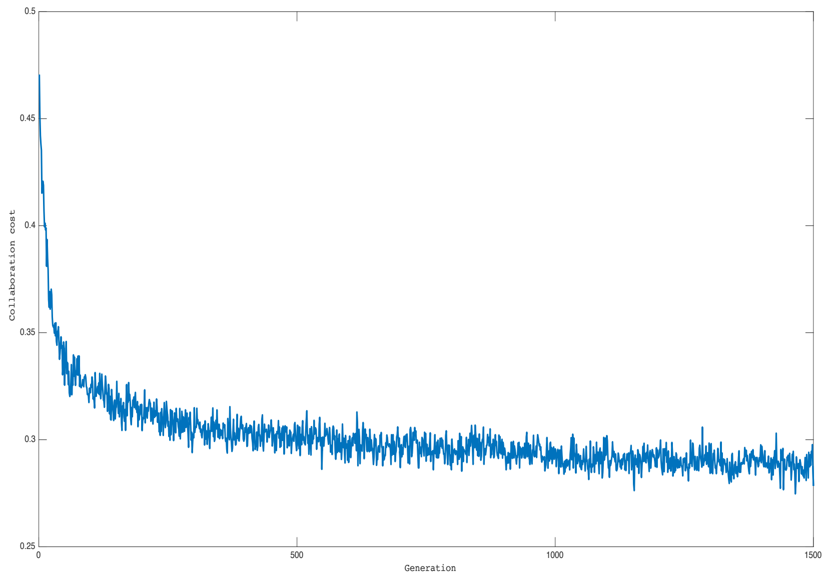


Figure 5.7 Problem 5: Hamming distance

In order to analyse the influence of those parameters on the quality of the solutions, different parameter values were considered for the tournament size, the crossover probability and the mutation rate. Regarding the tournament size, n_{ts}, the set of parameters considered was formed by [10% 20% 30% 40%].

The results shown in Figure 5.8 correspond to Problem 5, a relatively complex version of the TFPSN. In this case, one can observe that the GA obtains better results, as measured by lower collaboration costs, when the tournament size is high. This means that the algorithm favours selection pressure, as better solutions tend to be chosen. This is shown in Figure 5.9 in terms of an inverse relationship of the tournament size with the Hamming distance: high tournament size results in lower diversity, as measured by the Hamming distance.

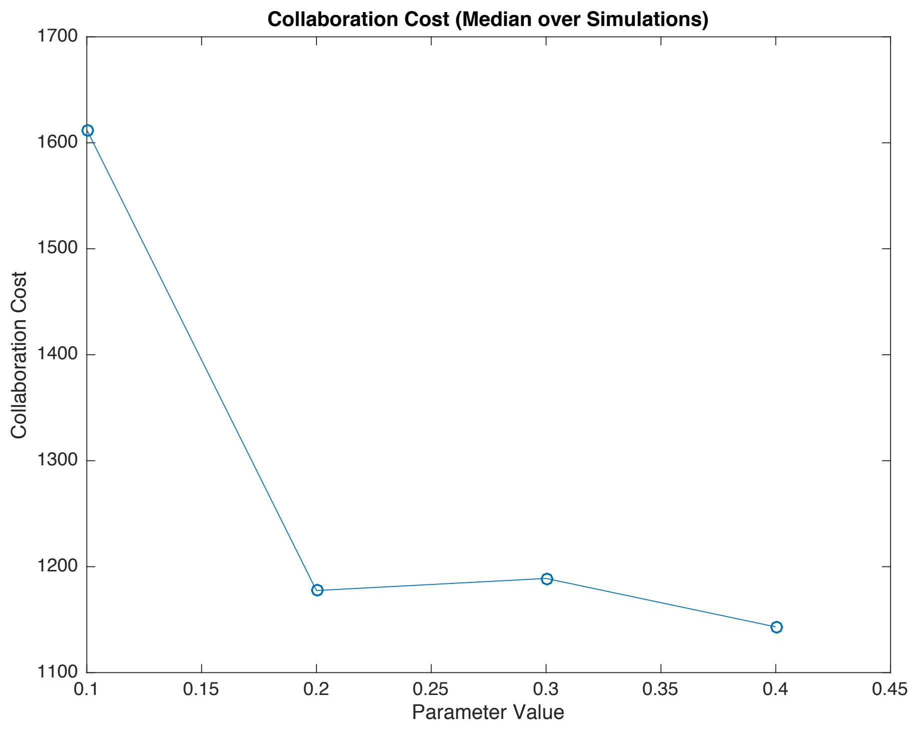


Figure 5.8 Problem 5: Best cost for different Tournament sizes

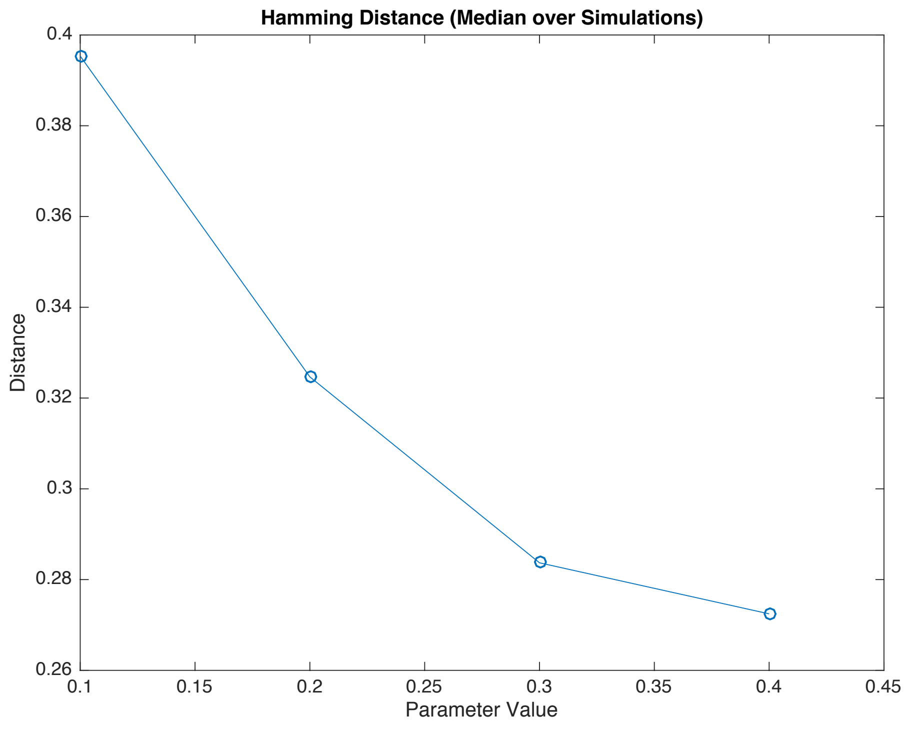


Figure 5.9 Problem 5: Hamming distance vs. Tournament size

Next, we analyse different parameter values for p_c, the crossover probability. Specifically, the following values were considered: [0.5; 0.65; 0.85; 0.95]. The results for Problem 5 are shown in Figure 5.10. In this case, we observe that the best results, in terms of minimum collaboration cost, are obtained with crossover probabilities in the lower range of those considered in the literature, that is, between 0.6 and 0.7. This means that the probability of inheriting the corresponding genes from the parents is relatively low. As a result, the selection pressure is reduced.

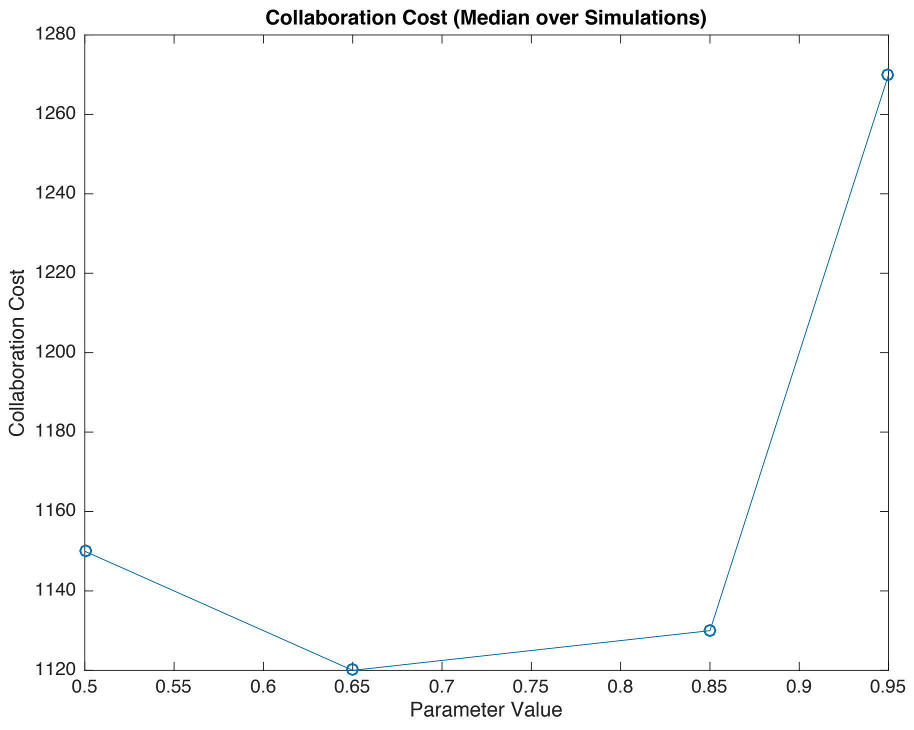


Figure 5.10 Problem 5: Best cost for different Crossover rates

Figure 5.11 shows the corresponding Hamming distance. As expected, the degree of diversity amongst the candidate solutions is lower the lower the probability of cross-over.

Finally, the mutation rate is analysed. In this case, the range of values considered for the probability of mutation, p_m, consisted of the following: [0.05; 0.15; 0.25; 0.35]. The results corresponding to Problem 5 are shown in Figure 5.12 where we can observe that the lower the mutation rate, the lower the collaboration cost. Figure 5.13 shows the corresponding Hamming distance. For low values of the mutation rate, the level of diversity is also low. This seems to suggest that the algorithm does not favour the exploration of alternative parts of the solution space.

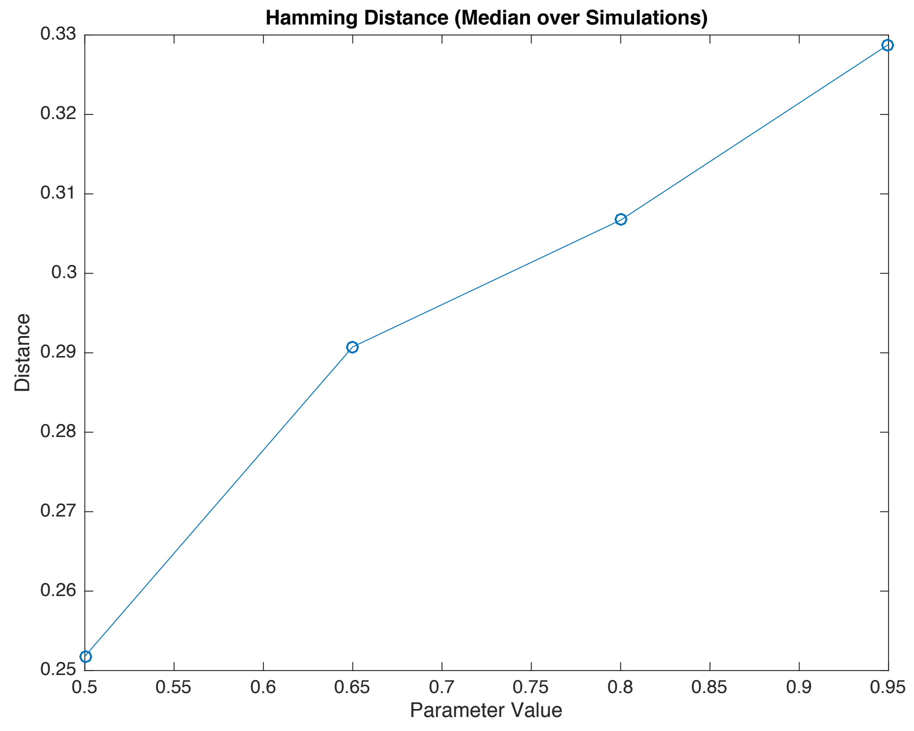


Figure 5.11 Problem 5: Hamming distance vs. Crossover rate

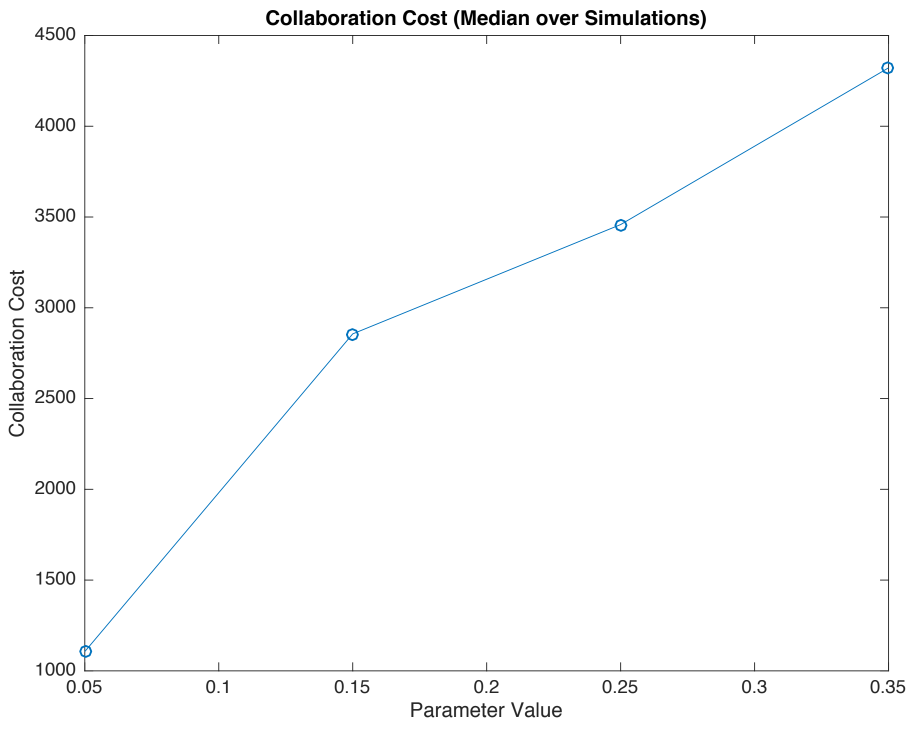


Figure 5.12 Problem 5: Best cost for different Mutation rate

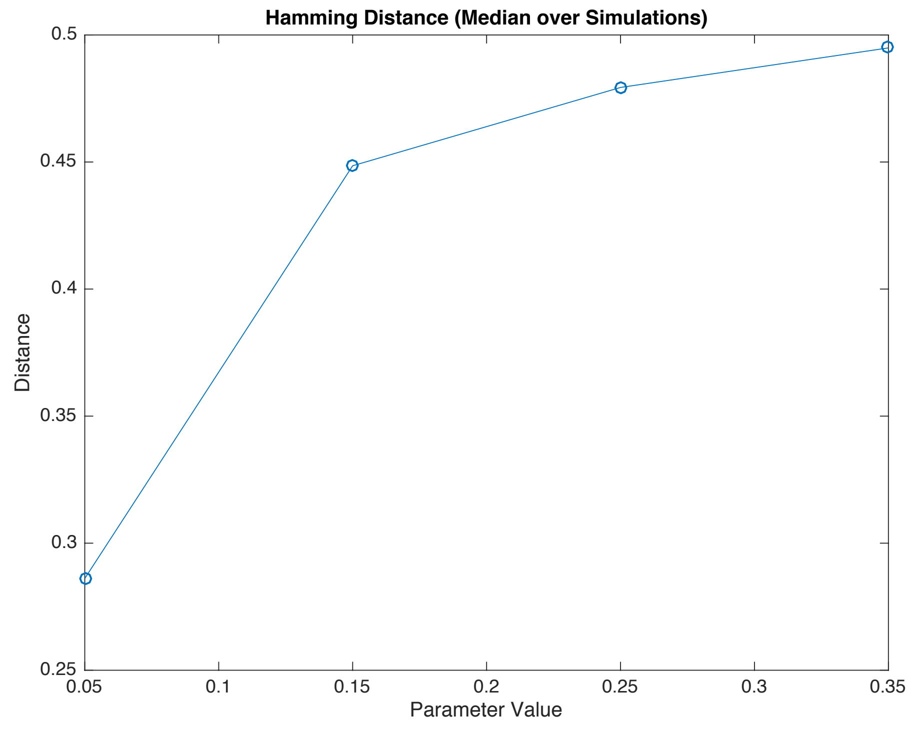


Figure 5.13 Problem 5: Hamming distance vs. Mutation rate

In summary, the GA is able to find relatively good solutions to the TFPSN for versions of the problem which are of a relatively moderate complexity. As expected, the performance deteriorates when the size of the problem is high. The results could be improved, for the particular case analysed here, when more selection pressure is incorporated into the genetic algorithm.

5.3 Evaluation of the QEA

This section analyses the performance of the QEA applied to the synthetic dataset described in Section 5.1. A key element of the QEA is the process to updating the Q-bits. This is very relevant as it will drive the search carried out by the algorithm and thus the process for getting closer to fitter solutions.

As in the case of the GA, the size of the population was set to 50 individuals and the maximum number of generations to 1500. A total number of 20 simulations were performed for each version of the TFPSN analysed.

5.3.1 Baseline Specification

First, a baseline version of the QEA was considered. There are two main components that characterise the algorithm, namely, the parameterisation of the rotation angle and the choice of an attractor, that is, the migration strategy. Regarding the latter, the baseline version of the QEA adopts a local approach whereby each Q-bit individual has its own attractor and no information is shared across individuals.

Regarding the parameterisation of the rotation angle, the specific values assigned depend on the problem considered and some experimentation might be required. Following Han et al. (2002), in this version of the QEA, the rotation angle is updated only in two cases as described in Table 5.4.

The typical values considered in the literature for the rotation parameter, denoted by θ, are in the range of $(0.001, 0.08)$. However, as noted above, this critically depends on the particular problem addressed. Hence, in order to obtain reference values for the TFPSN, a detailed analysis has been performed.

Firstly, it is considered the situation where the Q-gate rotation angle is fixed, that is, $\theta_1 = \theta_2$. For the numerical analysis, the version corresponding to Problem 4 of the TFPSN as described in Section 5.1 is taken as reference. This model consists of pools of 100 experts with the objective of forming teams consistent of, at least, 5 different skills.

Case	x_{ji}	b_{ji}	$f(x_j) \geq f(b_j)$	$\Delta\theta$
1	0	0	False	0
2	0	0	True	0
3	0	1	False	$\theta_1\pi$
4	0	1	True	0
5	1	0	False	$-\theta_2\pi$
6	1	0	True	0
7	1	1	False	0
8	1	1	True	0

Table 5-4 Look-up table for baseline QEA with rotation parameters θ_1 and θ_2

The results summarised in Table 5.5 show a high dependency of the QEA on the specific parameter considered for the rotation angle θ. The best minimum cost achieved by the algorithm indicates that the best solution is obtained with a relatively high parameter value. In the particular case considered here it is $\theta = 0.071$. However, relatively good results are obtained also with a much lower value of the rotation angle, for instance $\theta = 0.01$. Hence, a further analysis of this parameter is required.

Rotation Angle θ	0.001	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
Minimum Cost	89.94	2.89	4.30	5.91	22.88	22.07	37.97	2.18	47.71

Table 5-5 Problem 4: Minimum cost with rotation parameters $\theta_1 = \theta_2$

First, it must be noted that the objective of the rotation update is to change the direction of the Q-bit, in order to increase the chance of observing a particular state and, thus, obtaining fitter solutions. This parameter also influences the speed of convergence. Figure 5.14 illustrates this feature of the algorithm for different values of the rotation parameter. Specifically,

$$\theta = \{0.001,\ 0.01,\ \ 0.02,\ \ 0.03,\ \ 0.04\ \ 0.05\ \ 0.06\ \ 0.07\ \ 0.08\}.$$

The results obtained show that the speed of convergence increases the higher is the rotation angle. For relatively low rotation values, such as $\theta = 0.001$, the algorithm progresses relatively slowly. In contrast, for high values, premature convergence is more likely.

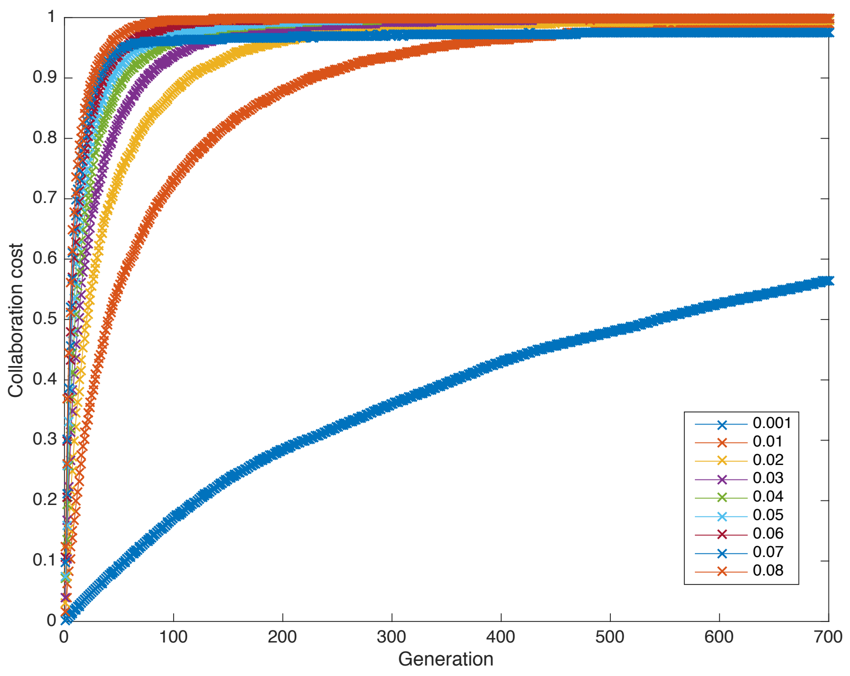


Figure 5.14 Problem 4: Convergence QEA for different rotation parameters

The next step in the analysis is to consider the case where both parameters differ, that is, $\theta_1 \neq \theta_2$. The objective is to allow for more flexibility in the direction of the rotation angle. Figure 5.15 shows the results for different combinations of both parameters θ_1 and θ_2. Figure 5.16 represents the corresponding contour curves.

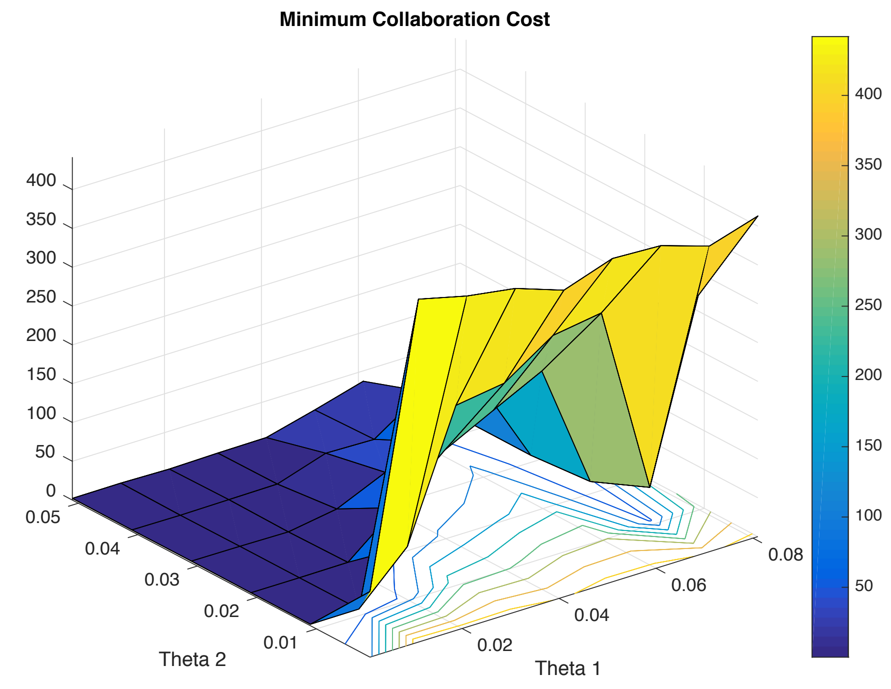


Figure 5.15 Problem 4: Minimum cost for varying rotation parameters θ_1 and θ_2

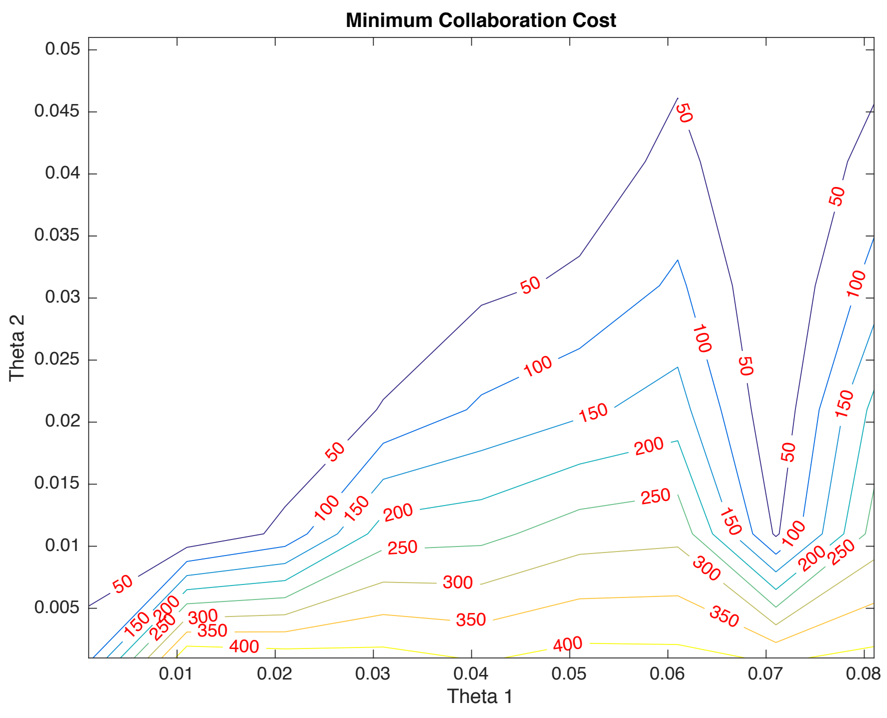


Figure 5.16 Problem 4: Contour plot for varying rotation parameters θ_1 and θ_2

The results confirm that the choice of the parameter values for the rotation angle is very relevant for the outcome of the QEA. In the specific version of the TFPSN considered here, the best results measured in terms of best minimum collaboration cost correspond to lower values of the rotation parameter θ_1 and higher of the parameter θ_2.

The behaviour of the QEA is, however, not uniform in the sense that there are also other areas of the parameter space where lower costs can also be achieved. Table 5.6 provides the specific numerical values in this regard. For instance, the best overall solution is achieved when $\theta_1 = 0.07$ and $\theta_2 = 0.05$. In this case, the collaboration cost for Problem 4 of the TFPSN is 0.3, which is significantly lower than the result obtained with the version of the genetic algorithm of Section 5.2. In that case, the best minimum cost obtained was 1.81.

θ_2 \ θ_1	0.001	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
0.001	84.5	441.8	426.5	416.8	395.1	416.5	413.7	393.7	413.0
0.01	1.9	2.1	63.3	225.6	234.9	276.9	286.5	42.4	270.3
0.02	2.0	1.9	2.4	53.4	108.2	140.0	171.2	9.4	214.9
0.03	2.3	1.9	1.1	11.2	39.0	58.8	109.3	2.7	120.7
0.04	2.0	1.5	2.2	2.9	23.2	21.6	63.9	3.0	67.3
0.05	2.2	2.5	1.1	2.3	2.9	19.1	36.6	0.3	29.9

Table 5-6 Problem 4: minimum cost varying rotation parameters θ_1 and θ_2

5.3.2 Time-varying Q-Gate Rotation Angles

From the analysis above, one can conclude that the baseline version of the QEA is able to achieve good results provided that a careful choice of the rotation parameters is carried out. However, the parameter in the look-up table was kept fixed throughout the entire execution of the algorithm. This could constraint the performance of the QEA.

A possible way to overcome these limitations is through the use of a dynamic mechanism for the rotation parameter. In this regard, one could take an interval $(\theta_{min}, \theta_{max})$, so that the rotation angle is allowed to vary within that range according to some pre-defined rule. For instance, the following expression could be considered:

$$\theta_j^t = \theta_{max} - (\theta_{max} - \theta_{min})D_j^t,$$

where D_j^t is a variable that measures the degree to which the changing rotation angle varies between the lower and upper bounds. As described in Wang et al. (2014), one simple option is to adjust the rotation angle by applying a factor defined as the ratio of t, the number of the current generation, over t^{max}, which denotes the total number of generations considered in the QEA. That is,

$$D_j^t = \frac{t}{t^{max}}.$$

In order to assess the adapting approach for the rotation parameter, we consider the case where $\theta_1 \in (0.01, 0.015)$ and the parameter varies according to the rule describe above. The other rotation parameter, θ_2, is kept fixed at 0.35. Problem 4 is chosen as the reference model. The results, shown on the first row of Table 5.7, indicate that the algorithm achieves a good solution in terms of low costs. Specifically, the minimum collaboration cost obtained is 0.55, which is below those obtained with the standard parameterisation of the Genetic Algorithm presented in section 5.2 which, for convenience, are reproduced in Table 5.7.

TFPSN Problem 4	Collaboration Cost (sum of distances)				Hamming Distance	Team Size / Skills
	Min	Max	Median	Std. Dev.		
QEA	0.55	2.92	2.59	0.79	0.04	5 / 5
GA	1.81	4.43	2.82	0.58	0.18	5 / 5

Table 5-7 Problem 4: minimum cost with varying rotation as in Wang et al. (2014)

In terms of the dynamics, Figure 5.17 shows that the algorithm approaches a solution relatively quickly and remains in that region thereafter. Hence, it is important that the algorithm focus the search on a good solution since, as the algorithm progresses, the size of the rotation angle decreases and the algorithm gets stuck in that region , as shown in Figure 5.18.

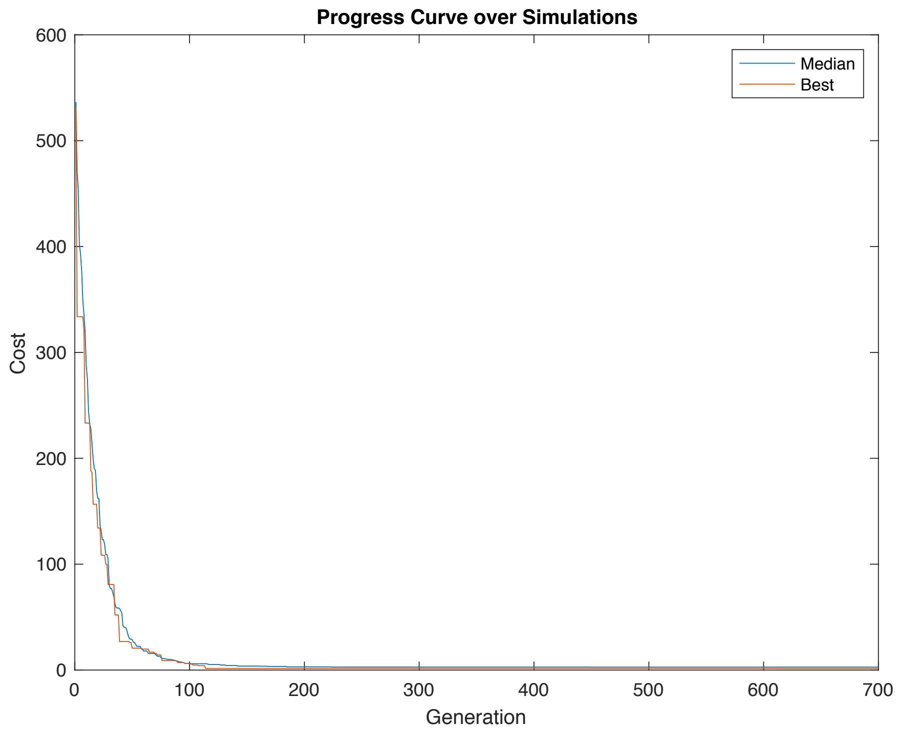


Figure 5.17 Problem 4: Progress Curve QEA with time-varying θ_1 and fixed θ_2

The evolving approach above for the rotation angle is relatively simple and mechanistic. In particular, it did not take into account information on the evolution of the search itself. In order to investigate alternative strategies for the evolution of the rotation parameter, we follow Lahoz-Beltra (2016). In this set-up, the factor that controls the degree to which the changing rotation angle varies between the lower and upper bounds is defined as:

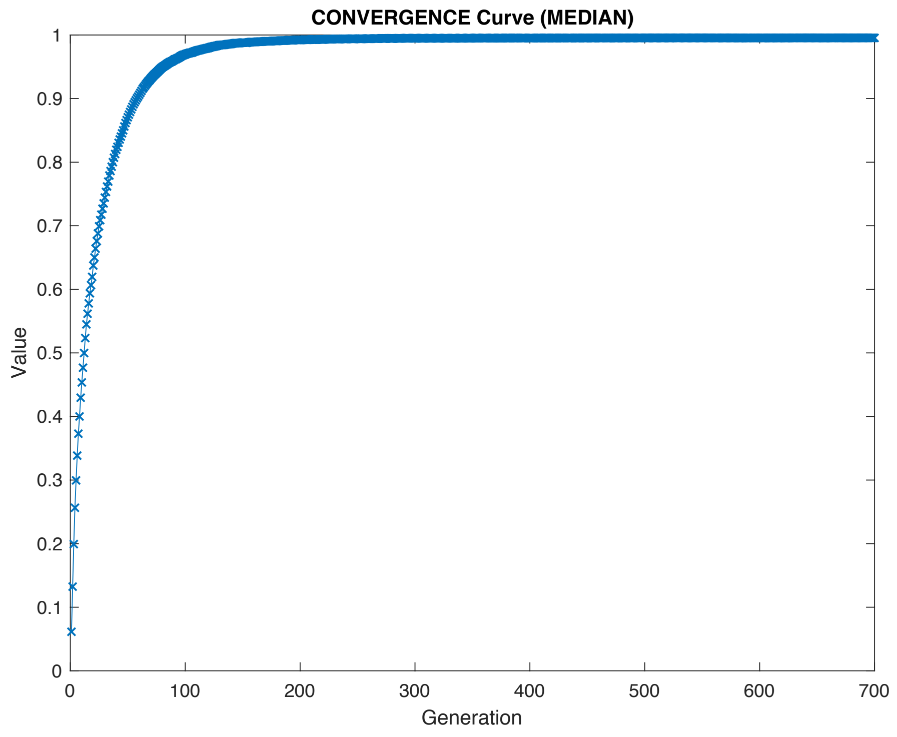


Figure 5.18 Problem 4: Convergence QEA with time-varying θ_1 and fixed θ_2

$$D_j^t = \frac{\left|F\left(\boldsymbol{x}_j^t\right) - F\left(\boldsymbol{b}_j^t\right)\right|}{F^t}.$$

where

$$F^t = \max\left\{1, \max_{\forall j \in n}\left\{\left|F\left(\boldsymbol{x}_j^t\right) - F\left(\boldsymbol{b}_j^t\right)\right|\right\}\right\}.$$

The objective of this evolving approach is to progressively reduce the size of the rotation angle as the candidate solutions approach their respective attractors. The results are shown in Table 5.8 together with those of the GA for reference. We observe that the minimum cost for the particular simulations performed is lower than in the case of a GA.

When comparing with the results using the mechanistic rule of Wang et al. (2014), we observe that the QEA reached a best minimum cost with this mechanistic rule than with the adaptive method of Lahoz-Beltara (2016). However, the median cost across simulations is lower in the latter.

TFPSN Problem 4	Collaboration Cost (sum of distances)				Hamming Distance	Team Size / Skills
	Min	Max	Median	Std. Dev.		
QEA	1.33	2.50	1.94	0.38	0.04	5 / 5
GA	1.81	4.43	2.82	0.58	0.18	5 / 5

Table 5-8 Problem 4: minimum cost with varying rotation as in Lahoz-Betra (2016)

From the discussion above, parameter tuning is an important part in the design of the QEA as it affects the efficacy of the search process.[18] In fact, most of the effort in designing these algorithms is spent in parameter tuning. It is a difficult optimization problem in itself as it is usually poorly structured, ill-defined and complex in nature. In addition, the best set of parameter values can be guaranteed to be found only after exhaustive search in the entire parameter-space, however, such a strategy may not be feasible in practice due to the large amount of resources and time consumed.

5.3.3 Analysis of Convergence

In the analysis presented so far, the QEA migration strategy (choice of attractor) was defined local in the sense that each Q-bit individual had its own attractor. Accordingly, when a new attractor b_i is chosen in the search space, the corresponding Q-bit is slightly moved toward this point until a better solution is found. The question then is what if not better solution is found during this move. In this case, the algorithm is trapped and converges prematurely to this point. Hence, the use of a local attractor strategy may cause premature convergence where the algorithm is stuck in a local optimum. The only opportunity for an individual to escape from this attractor is that a mechanism is in place so that it can be replaced with a better attractor produced elsewhere. Otherwise, it is possible that the choice of a good but sub-optimal attractor is irreversible. The probabilistic model becomes unable to produce solutions different from the attractor and, therefore, the QEA can be trapped. Hence, the bad choice of an attractor can quickly become irreversible.

QEAs are prone to prematurely converge, suffering mostly by the phenomenon of hitchhiking as experimentally shown in Platel et al. (2007). Hitchhiking corresponds to the increase in

[18] In general, the parameter tuning methods have been categorized as Sampling Methods, Model Based Methods, Screening Methods and Meta-Evolutionary Methods. These approaches are described in Mani (2017).

frequency of a "bad" allele at a given locus in the population due to the presence of nearby highly fit alleles on the same chromosomes. As a consequence, the eventual better alleles at the same locus (as the hitchhiking allele) tend to disappear in the population and there is no way for the evolutionary process to retrieve them.

Following Lee and Lin (2017), one way of quantifying possible premature convergence of the algorithm is by calculating the mean square deviation (MSD) of each individual attractor with respect to the global attractor. This measure is defined according to the following expression:

$$MSD^t = \frac{1}{N}\sum_{j=1}^{n}\left(\frac{F\left(\boldsymbol{b}_j^t\right) - F(\boldsymbol{b^t})}{\boldsymbol{F}^t}\right)^2,$$

where

$$\boldsymbol{F}^t = \max\left\{1, \max_{\forall j \in n}\left\{\left|F\left(\boldsymbol{b}_j^t\right) - F(\boldsymbol{b^t})\right|\right\}\right\}.$$

The results shown in Figure 5.19 corresponding to Problem 4 of the TFPSN indicate that the algorithm converges relatively quickly towards the global optimum. This could be a sign of premature convergence.

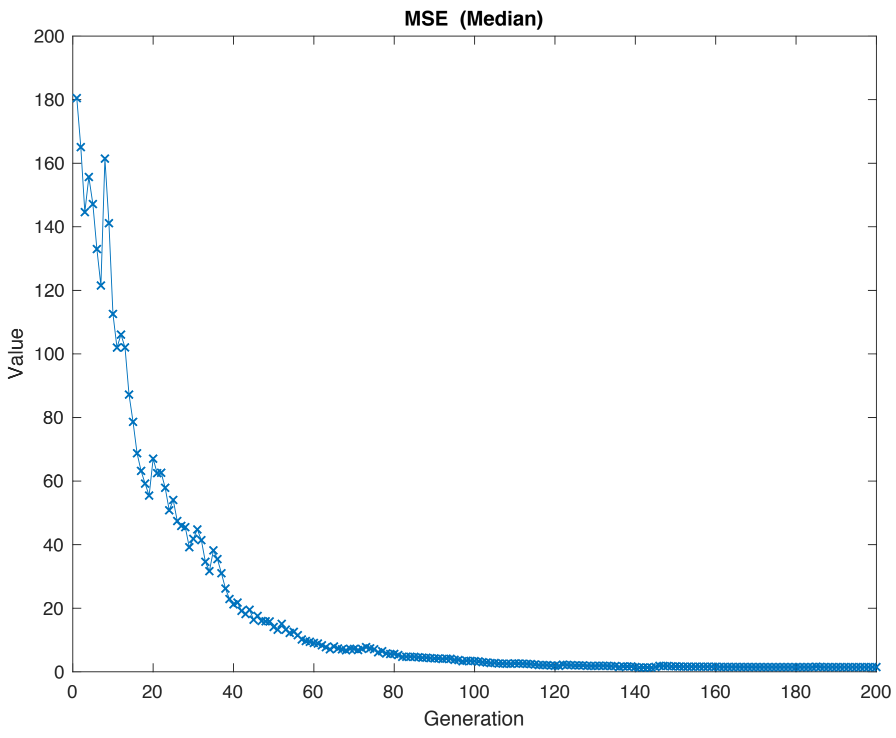


Figure 5.19 Problem 4: MSE Curve for QEA with time-varying θ_1 and fixed θ_2

In order to further analyse this extent, we compute the 75% and 25% quantiles corresponding to the empirical distribution of the attractors at each point in time (generation). Figure 5.20 shows the results which again may suggest the existence of premature-convergence. In this case, it would be advisable to apply some mechanism to scape possible local optima.

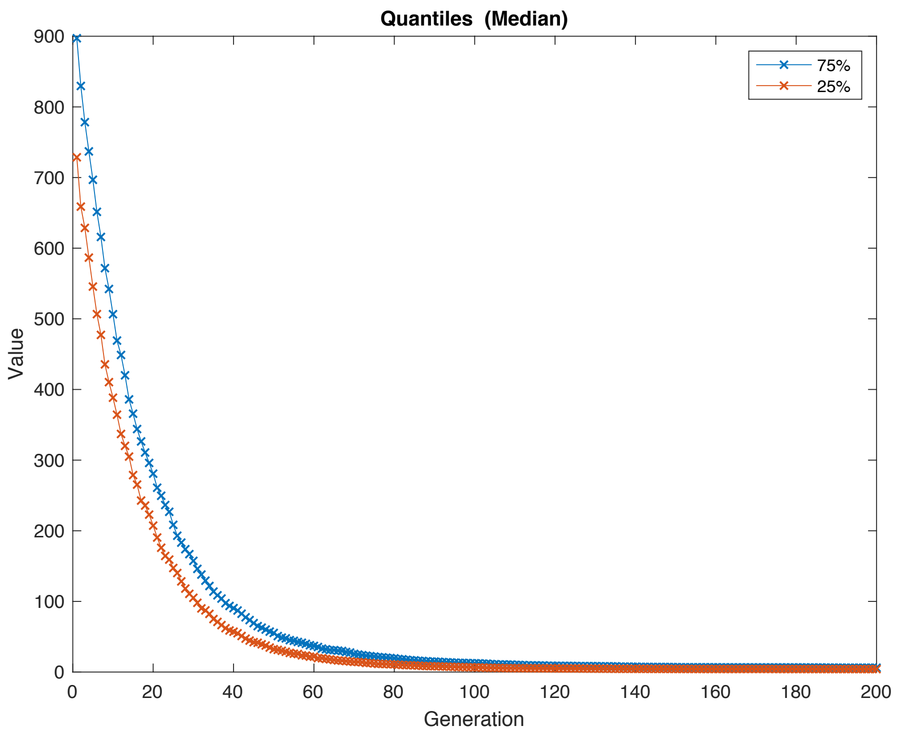


Figure 5.20 Problem 4: Quantiles for QEA with time-varying θ_1 and fixed θ_2

5.3.4 Alternative Attraction Strategies

In order to improve the performance of the QEA, and address potential premature convergence, we would consider alternative migration strategies. One possibility is to consider a global mechanism where a single attractor is used by all individuals in the population. In this case, the information about the search space collected during evolution is not kept at the individual level but continuously renewed and periodically shared among the whole population. Thus, eventual decision errors do not have persistent consequences. This mechanism is expected to lead to a smoother and more efficient long term exploration of the search space.

The results have been computed for all versions of the TFPSN and summarised in Table 5.9. In all cases considered, the QEA has been able to form teams with exactly the same size as the number of required skills. This is in contrast with the results obtained with the baseline GA where, for the more complex instances of the TFPSN, the teams corresponding to the best solutions were relatively large. Specifically, for Problem 5 the ratio of size to skill was 52/30 and 32/5 for Problem 6.

TFPSN		Best Collaboration Cost (sum of distances)				Hamming Distance	Team Size / Skills
Problem	Dimension	Min	Max	Median	Std. Dev.		
1	50-10	19.16	25.78	22.14	2.7	0.07	10/10
2	50-5	0.79	2.65	1.05	0.81	0.03	5/5
3	100-20	111.03	125.84	120.19	5.03	0.05	20 / 20
4	100-5	0.29	1.84	1.05	0.46	0.03	5/5
5	300-30	271.94	296.57	286	6.95	0.04	30/30
6	300-5	0.14	2.04	0.87	0.62	0.02	5/5

Table 5-9 Results for QEA with global attractor and adaptive parameters θ_1*and* θ_2

When analysing the results in terms of best minimum collaboration cost, it is interesting to note the following: For Problems 1 and 2, which can be considered as "easy" problems, both the QEA and the GA achieve the same solution. However, as the complexity of the TFPSN increases, the QEA provides better results. Figure 5.21 shows the ratio of best minimum cost between the QEA and the GA. For Problem 3, which could be considered still relatively easy, the ratio is 96%. However, for really complex versions of the TPPSN, the improvement shown by the QEA is dramatic. For instance, Problem 6, the costs corresponding to the best team obtained by the QEA is 0.1% that of the best minim cost achieved by the reference GA.

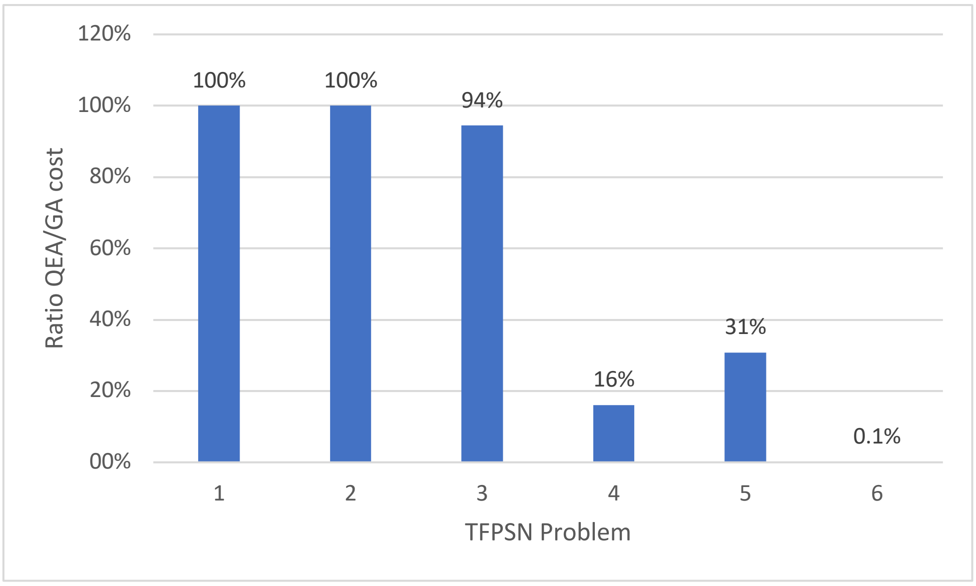


Figure 5.21 Best collaboration cost: comparison QEA vs. GA

The results presented above show the importance of sharing information across the population. In this section, we have implemented a global migration strategy. This means that at each generation, the overall best solutions is transmitted to all individuals that use it as an attractor. However, if a suboptimal solution is propagated to the global level, then this solution starts to attract the entire population. As long as no better solution is found, all the probabilistic models converge towards this global attractor. The use of an adaptive Q-gate rotation mechanism seems to help avoid this type of situations, making the QEA a competitive solution mechanism for hard combinatorial optimisation problems.

Chapter 6 Concluding Remarks and Future Research

The present work studies the application of a class of evolutionary algorithms to solving the Team Formation Problem in Social Networks. Team formation is a well-known and widely studied combinatorial optimisation problem. Essentially, it entails finding a team of experts that not only cover the required skills of a given task, but also the members can work effectively with each other. The social network is modelled as a graph where each node represents an expert, with one or more skills, and each edge is weighted by the direct collaboration cost between the two corresponding experts. The problem has been shown to be complex, belonging to the NP-hard class. Hence, no polynomial time algorithm exists that finds the optimal solution for all specific instances of this problem.

The approach considered here is inspired in the quantum processes of superposition. QEAs use Q-bits to represent the individuals, and searches for the optimum by observing the quantum states. It can work with small population sizes without running into premature convergence. A feature of QEAs is their ability to explore the search space even with a small number of individuals and to properly balance between exploration and exploitation.

The numerical analysis shows that the version of the QEA used in this work was able to solve relatively simple instances of the TFPSN in line with the performance of the reference GA. Indeed, for more difficult versions of the problem, the QEA yielded significantly better results. Accordingly, the algorithm was able to achieve solutions (form teams of experts) that not only satisfied the required skills but also produced combinations with much lower collaboration costs than the reference GA. These results are particularly interesting as the QEA has only one evolutionary parameter, namely the gate rotation angle, versus the three evolutionary operators of the baseline GA: tournament size, crossover and mutation rate.

One key element for the successful performance of the QEA was precisely the parameterisation of the gate rotation angle. A detailed analysis was conducted and different methods considered. The use of a time-varying approach whereby the magnitude of the rotation was made dependent on the distance of the solution to the attractor was shown to generate the best results. All in all, this work has provided evidence that the QEA is a feasible candidate to tackling difficult combinatorial optimisation problems.

The analysis conducted in this work was based on synthetic datasets. Hence, a possible extension would consist of conducting an empirical investigation using some reference data sets. Research experiments in this area are typically performed on collaboration networks databases such as IMD, that considers links amongst film actors, and DBLP, that contains collaborations amongst scientists.

Certainly, the baseline QEAs has some limitations and the application to difficult problems as the one considered here may require further extensions. As discussed in this work, QEAs are prone to premature convergence when the algorithm is trapped in a local optimum. This could be linked to the fact that the baseline QEA is an elitist algorithm. The exploration of the search space is driven by attractors corresponding to the best solution found either at the individual, local, or global level. If a suboptimal solution is propagated to the global level, then this solution starts to attract the entire population. As long as no better solution is found, all the probabilistic models converge towards this global attractor. The probabilistic model becomes unable to produce solutions different from the attractor and, therefore, the QEA can be trapped. Hence, the bad choice of an attractor can quickly become irreversible.

Accordingly, one aspect of the algorithm that could be enhanced is related to the structure of the population and how the information is shared across individuals or groups of individuals. For instance, the use of combinatorial graphs that limit the speed and the way information spreads could allow competing solutions to have more time to mature. Bryden et al. (2006) show that the use of graphs is a computationally inexpensive method of obtaining a globally acceptable level of trade-off between exploration and exploitation. There are two main approaches to modelling decentralized populations which can be summarised as follows:

(1) Coarse-grained algorithms (also known as Distributed Evolutionary Algorithms) are characterized by partitioning of the population into several subpopulations or islands, each of which runs independently and where individual exchanges among the islands occur with a given frequency;

(2) Fine-grained or Cellular Evolutionary Algorithms where, in the basic setup, individuals are placed on a toroidal grid. Every individual has a neighbourhood, and an individual can only interact with individuals belonging to its neighbourhood. An individual has its own pool defined by neighbouring individuals and, at the same time, an individual can belong to several pools. The connections among neighbourhoods help the algorithm exploit possible solutions, and the overlapped small neighbourhoods help the algorithm explore the search space. Alba and Dorronsoro (2008) provide a detailed analysis of this type of algorithms. Dorronsoro et al. (2013) and Burguillo et al. (2018) present methods to enhance their performance.

Cellular structures have been applied to QEAs showing good results. Tayarani et al. (2008), Qi (2015) and Mani (2017) are also relevant examples within this line of research.[19] Hence, the application of these other approaches to the TFPSN constitutes an interesting research avenue.

[19] Gupta et al. (2017) proposed a single-population fine-grained approach that is suited for massively parallel computations. The authors apply this approach to community detection in social networks.

Bibliography

Alba, E. and Dorronsoro, B. 2008. Cellular Genetic Algorithms. Operations Research/Computer Science Interfaces Series. Springer-Verlag.

Agustín-Blas LE, Salcedo-Sanz S, Ortiz-García EG, Portilla-Figueras A, Pérez-Bellido AM, Jiménez- Fernández S. 2010. Team formation based on group technology: a hybrid grouping genetic algorithm approach. Comput Oper Res 38:484–495

Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2012) Online team formation in social networks. In: Proceedings of ACM International Conference on World Wide Web (WWW'12), pp 839–848

Arpaia, P., Maisto, D. and Manna, C. 2011. A Quantum-inspired Evolutionary Algorithm with a competitive variation operator for Multiple-Fault Diagnosis. Applied Soft Computing 11 (2011) 4655–4666

Awal, G. and Bharadwaj, K. 2014. Team formation in social networks based on collective intelligence – an evolutionary approach. Applied Intelligence, Vol. 41, pp. 627-648.

Awal G. and Bharadwaj K. 2018a. Harnessing Collective Intelligence Through Pattern Mining in Social Computational Systems. Handbook of Research on Pattern Engineering System Development for Big Data Analytics. IGI Global, pp 91-110.

Awal G. and Bharadwaj K. 2018b. Constrained Team Formation Using Risk Estimation Based on Reputation and Knowledge. In: Saeed K., Chaki N., Pati B., Bakshi S., Mohapatra D. (eds) Progress in Advanced Computing and Intelligent Engineering. Advances in Intelligent Systems and Computing, vol 564. Springer, Singapore.

Bäck, T., Fogel, D. and Michalewicz, Z. 2000a. Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics Publishing, Bristol, UK

Bäck, T., Fogel, D. and Michalewicz, Z. 2000b. Evolutionary Computation 2: Advanced Algorithms and Operators. Institute of Physics Publishing, Bristol, UK

Basiri, J., Taghiyareh, F. and Ghorbani, A. 2017. Collaborative team formation using brain drain optimization: a practical and effective solution. World Wide Web (2017) 20:1385–1407

Baykasoglu, A., Dereli, T., and Das, S.. 2007. Project team selection using fuzzy optimization approach. Cybern. Syst., 38(2):155–185.

Beasley JE, Chu PC (1996) A genetic algorithm for the set covering problem. Eur J Oper Res 94:392–404

Boussaïd, I., Lepagnot, J. and Siarry, P. 2013. A survey on optimization metaheuristics. Information Sciences. 237: 82-117.

Burguillo J.C., Dorronsoro B. 2018. Optimization Models with Coalitional Cellular Automata. In: Self-organizing Coalitions for Managing Complexity. Emergence, Complexity and Computation, vol 29. Springer.

Chhabra, M., Das, S. and Szymanski, B. 2012. Team Formation in Social Networks. In Computer and Information Sciences III. 27th International Symposium on Computer and Information Sciences. Editors: Gelenbe, Lent. Springer.

Crepinsek, M., Liu, S. and Mernik, M. 2013. Exploration and exploitation in evolutionary algorithms: A survey. ACM Computing Surveys (CSUR). Volume 45 Issue 3, June 2013. Article No. 35.

Datta, S., Majumder, A. and Naiu, K. 2012. Capacitated Team Formation Problem on Social Networks. Pre-print. ArXiv:1205.3643.

De Jong, K.A. 2006. Evolutionary Computation: A Unified Approach. The MIT Press, Cambridge, MA.

Dorn C, Skopik F, Schall D, Dustdar S. 2011. Interaction mining and skill-dependent recommendations for multi-objective team composition. Data Knowledge Engineer 70(10):866–891.

Dorronsoro B., Burguillo J.C., Peleteiro A., Bouvry P. (2013) Evolutionary Algorithms Based on Game Theory and Cellular Automata with Coalitions. In: Zelinka I., Snášel V., Abraham A. (eds) Handbook of Optimization. Intelligent Systems Reference Library, vol

38. Springer, Berlin, Heidelberg

Du, K.-L. and Swamy, M.N.S. Search and Optimization by Metaheuristics. Chapter 17. Springer International Publishing Switzerland.

Duhaime, E., Olson, G. and Malone, T. 2015. Broad participation in collective problem solving can influence participants and lead to better solutions: Evidence from the MIT Climate CoLab. MIT Center for Collective Intelligence. Working Paper 2015-001.

Dykhuis, N., Cohen, P. and Chang, Y. 2013. Simulating Team Formation in Social Networks. SocialCom / PASSAT / BigData / EconCom / BioMedCom 2013.

Easley, D. A. and Kleinberg, J. M. 2010. Networks, Crowds, and Markets - Reasoning About a Highly-Connected World. Cambridge University Press, Oxford, UK.

Eiben, A. and Smith, J. 2003. Introduction to Evolutionary Computing. Springer, Berlin.

El-Asmawi, W., Ali, A. and Tawhid, M. 2018. An improved particle swarm optimization with a new swap operator for team formation problem. Journal of Industrial Engineering International.

Engelbrecht, A. P. 2007. Computational Intelligence: An Introduction. 2nd edition. John Wiley & Sons.

Esgario, J., da Silva, I. and Krohling, R. 2019. Application of Genetic Algorithms to the Multiple Team Formation Problem. Arxiv 1903.03523. Accessed on 8 March 2019.

Fathian, M. Saei-Shahi, M. and Makui, A. 2018. A new optimisation model for reliable team formation problem considering experts' collaboration network. 586 IEEE Transactions on Engineering on Management, vol. 64, no. 4, November 2017

Gowers, T. and Nielsen, M. 2009. Massively collaborative mathematics. Nature 461 879–881.

Gupta, S., Mittal, S., Gupta, T., Singhal, I., Khatri, B., Gupta, A. Kumar, N. 2017. Parallel quantum-inspired evolutionary algorithms for community detection in social networks. Applied Soft Computing 61: 331–353

Han, K.,-H. and Kim, J.-H. 2000. Genetic quantum algorithm and its application to combinatorial optimization problem, in: Proceedings of the Congress on Evolutionary

Computation, vol. 2, IEEE, 2000, pp. 1354–1360.

Han, K.-H. and Kim, J.-H., 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Trans. Evolut. Comput. 6-580.

Hauschild, M. and Pelikan, M. 2011. An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1: 111–128.

Holland, J.H. 1975. Adaptation in Natural and artificial Systems. University of Michigan Press, Ann Arbor.

Kargar, M., An, A. 2011. Discovering top-k teams of experts with/without a leader in social networks. In: CIKM, pp. 985–994. ACM

Kargar, M., A. An, and M. Zihayat. 2012. Efficient bi-objective team formation in social networks, in Machine Learning and Knowledge Discovery in Databases. Springer, pp. 483–498.

Kargar, M., Zihayatm, M., An, A. 2013a. Finding Affordable and Collaborative Teams from a Network of Experts. Proceedings of the SIAM International Conference on Data Mining (SDM), 587—595.

Kargar, M., Zihayatm, M., An, A. 2013b. Affordable and collaborative team formation in an expert network. Department of Computer Science and Engineering, York University, Technical Report CSE-2013, vol. 1.

Klug, M. and Bagrow, J. 2016. Understanding the group dynamics and success of teams. Royal Society Open Science. 3:1600007.

Kyriakou, H. and Nickerson, J. 2014. Collective Innovation in Open Source Hardware. Stevens Institute of Technology.

Lahoz-Beltra, R. 2016. Quantum Genetic Algorithms for Computer Scientists. Computers 5, 24.

Lappas T, Liu K, Terzi E. 2009. Finding a team of experts in social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’09), pp 467–475

Lee, C. and Lin, B. 2017. Applications of the Chaotic Quantum Genetic Algorithm with Support Vector Regression in Load Forecasting. Energies, 10, 183.

Li, C., Shan, M. and Lin, S. 2015. On team formation with expertise query in collaborative social networks. Knowledge Information Systems. Vol. 42, pp.441–463.

Majumder, A., Datta, S., Naidu, K. 2012. Capacitated team formation problem on social networks. In: SIGKDD, pp. 1005–1013. ACM

Malone, T. and Bernstein, M. (Editors). 2015. Handbook of Collective Intelligence. MIT Press. Cambridge, MA.

Mani, N., Gursaran, and Mani A. 2017. Solving Combinatorial Optimization problems with Quantum inspired Evolutionary Algorithm Tuned using a Novel Heuristic Method. Mimeo

Mani, N., Srivastava, S. and Mani, A. 2017. Design of cellular quantum-inspired evolutionary algorithms with random topologies. In Quantum Inspired Computational Intelligence: Research and Applications. Edited by Bhattacharyya, Maulik and Dutta. Elsevier Inc.

Manju, A. and Nigam, M.J. 2014. Applications of quantum inspired computational intelligence: a survey. artificial intelligence Review (2014) 42:79–156

Michelucci, P. and Dickinson, J. 2016. The power of crowds: Combining humans and machines can help tackle increasingly hard problems. Science. Volume 351, Issue 6268, pp. 32-33.

Moussaid, M., Garnier, S., Theraulaz, G. and Helbing, Dirk. 2009 Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds. Topics in Cognitive Science 1 (2009) 469–497

Nakayama, S., Imabeppu, T and Ono, S. (2006). Pair swap strategy in Quantum-inspired Evolutionary Algorithms. GECCO 06.

Narayanan, A. and Moore, M. 1996. Quantum-inspired genetic algorithms. Proceedings of the IEEE International Conference on Evolutionary Computation. Nagoya. Japan.

Newell, A. and Simon, H. 1972. Human Problem Solving. Prentice-Hall.

Nickerson, J. 2015. Collective Design: Remixing and Visibility. In J.S. Gero, S. Hanna (eds.), Design Computing and Cognition '14. Springer.

Morrison R.W., De Jong K.A. (2002) Measurement of Population Diversity. In: Collet P., Fonlupt C., Hao JK., Lutton E., Schoenauer M. (eds) Artificial Evolution. EA 2001. Lecture Notes in Computer Science, vol 2310. Springer, Berlin, Heidelberg

Pendharkar, P. 2013. Genetic learning of virtual team member preferences. Computers in Human Behavior. Vol. 29, pp. 1787–1798.

Pitchai, A., Reddy A. and Savarimuthu, N. 2016. Fuzzy based Quantum Genetic Algorithm for Project Team Formation. International Journal of Intelligent Information Technologies. Vol. 12, N. 1.

Platel, M., Schliebs, S. and Kasavov, N. 2007. A Versatile Quantum-inspired Evolutionary Algorithm. Manuscrpt. Knowledge Engineering and Research Institute (KEDRI).Auckland University of Technology, 2007.

Platel, M., Schliebs, S. and Kasavov, N. 2009. Quantum-Inspired Evolutionary Algorithm: A Multimodel EDA. IEEE Transactions on Evolutionary Computation. Vol. 13, N. 6., December 2009.

Rangapuram, S., Bühler, T. and Hein, M. 2015.Towards Realistic Team Formation in Social Networks based on Densest Subgraphs. Manuscript. arXiv:1505.06661v1.

Rowe, J. 2015. Genetic Algorithms. In Janusz Kacprzyk, Witold Pedrycz (Eds.) Handbook of Computational Intelligence. Springer-Verlag. pp 825-844

Shi, Y., Zhiyong Peng, Liang Hong, Qian Yu 2017. SoC-constrained team formation with self-organizing mechanism in social networks. Knowledge-Based Systems 138 (2017) 1–14.

Singh, V. and A. K. Gupta, A. 2009. From artificial to Collective Intelligence: Perspectives and Implications. 5th International Symposium on Applied Computational Intelligence and Informatics.

Simon, H. A. 1955. On a class of skew distribution functions. Biometrika 42, 425–440.

Strnad, D. and Guid, N. 2010. A fuzzy-genetic decision support system for project team formation. Applied Soft Computing, vol. 10, pp. 1178-1187.

Surowiecki, K. 2004. The Wisdom of Crowds, Anchor Books.

Tayarani, M. and Akbarzadeh, R. 2008. A Cellular Structure and Diversity Preserving Operator in Quantum Evolutionary Algorithms. In: Proceedings of IEEE Congress on Evolutionary Computation. 2670-2675, (2008).

Tarayani, M. and Akbarzadeh, R. 2014. Improvement of the Performance of the Quantum-inspired Evolutionary Algorithms: structures, population, operators. Evolutionary Intelligence. 7:219-239.

Teng, Y., Wang, J., and Huang, J. 2014. Team Formation with the Communication Load Constraint in Social Networks. Trends and Applications in Knowledge Discovery and Data Mining Volume 8643 of the series Lecture Notes in Computer Science pp 125-136

Theiner, G., Allen, C. and Goldstone, R. 2009. Recognizing group cognition. Cognitive Systems Research 11 (2010) 378–395.

Turing A. M. 1948. Intelligent Machines, Reprinted in Ince D.C. (ed) (1992) Mechanical Intelligence: Collected Works of A. M. Turing, pp 21–23, North-Holland

Turing A. M. 1950. Computing Machinery and Intelligence. Mind 59(236):433–460. Reprinted in Ince D.C. (ed) (1992) Mechanical Intelligence: Collected Works of A. M. Turing, North-Holland

Vajda, P., Eiben, A.E. and Hordijk, W. 2008. Parameter Control Methods for Selection Operators in Genetic Algorithms. In Parallel Problem Solving from Nature, 10th International Conference, Proceedings, volume 5199 of Lecture Notes in Computer Science, pages 620–630. Springer, 2008.

Wang, H., Li, L, Liu, J. Wang, Y. and Fu, Ch. 2014. Improved Quantum Genetic Algorithm in Application of Scheduling Engineering Personnel. Abstract and Applied Analysis Volume 2014, Article ID 164961.

Wang, W., Zhao, Z. and Ng, W. 2015. A Comparative Study of Team Formation in Social Networks. Database Systems for Advanced Applications Volume 9049 of the series Lecture Notes in Computer Science pp 389-404.

Wendin, Göran. 2019. Can Biological Quantum Networks Solve NP-Hard Problems? Advanced Quantum Technologies, vol. 2.

Wi H, Oh S, Mun J, Jung M. 2009. A team formation model based on knowledge and collaboration. Expert Syst Appl 36:9121–9134.

Whitley, D., Rana, S. and Heckerdorn, R. 1999. The Island Model Genetic Algorithm: On Separability, Population Size and Convergence. Journal of Computing and Information Technology. Vol. 1, pp. 33-47.

Xing, H., Liu, X., Jin, X., Bai, L. and Ji, Y. 2009. A multi-granularity evolution based Quantum Genetic Algorithm for QoS multicast routing problem in WDM networks. Computer Communications 32: 386–393.

Yang, Y. and Hu, H. 2013. Team formation with time limit in social networks, in Mechatronic Sciences, Electric Engineering and Computer (MEC), Proceedings 2013 International Conference on. IEEE, 2013, pp. 1590–1594.

Yingchareonthawornchai, S., Aporntewan, C. and Chongstitvatana, P. 2012. An Implementation of Compact Genetic Algorithm on a Quantum Computer. Ninth International Joint Conference on Computer Science and Software Engineering (JCSSE) .

Zamudio, E., Berdún, L. and Amandi, A. 2016. Social networks and genetic algorithms to choose committees with independent members. Expert systems with Applications. Volume 43, January 2016, Pages 261–270.

Zhang, G. 2011. Quantum-inspired evolutionary algorithms: a survey and empirical study. Journal of Heuristics. 17: 303-351.