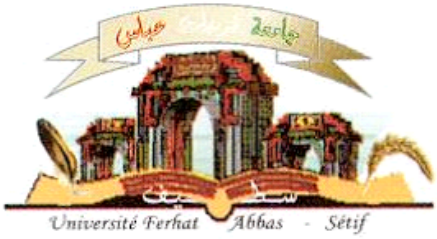
People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Ferhat Abbas Setif -1-



Thesis

Presented to the Faculty of Sciences
Computer Science department
to obtain the title of

PhD of Science

Specialty : COMPUTER SCIENCE

Defended by

Djaafar ZOUACHE

Bio-inspired Algorithms for Data Mining

Defended on April 11, 2016

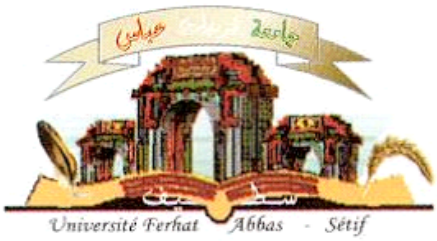
Jury:

President :	Abdellah KHABABA	Prof.	University of Ferhat Abbas Setif -1
Advisor :	Abdelouahab MOUSSAOUI	Prof.	University of Ferhat Abbas Setif -1
Examinators :	Salim CHIKHI	Prof.	University of Constantine 2
	Mohamed Tarek KHADIR	Prof.	University of Annaba
	Mustapha BOURAHLA	MCA.	University of M'sila
	Mohamed SAIDI	MCA.	University of Ferhat Abbas Setif -1

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ferhat Abbas de Sétif -1-



Thèse

Présentée à la Faculté des Sciences
Département d'Informatique
En vue de l'obtention du diplôme de

Doctorat en Sciences

Option : INFORMATIQUE

Par

Djaafar ZOUACHE

Thème

Fouille de données basée algorithmes bio-inspirés

Soutenu le: / /2015 devant le jury composé de :

Président :	Abdellah KHABABA	Prof.	Université de Sétif 1
Rapporteur :	Abdelouahab MOUSSAOUI	Prof.	Université de Sétif 1
Examinateurs :	Salim CHIKHI	Prof.	Université de Constantine 2
	Mohamed Tarek KHADIR	Prof.	Université d'Annaba
	Mustapha BOURAHLA	MCA.	Université de M'sila
	Mohamed SAIDI	MCA.	Université de Sétif 1

Acknowledgments

I would like to express my deep gratitude to my supervisors, Pr. Abdelouahab MOUSSAOUI, for his invaluable advice, time and support throughout this research work. This thesis would not have been possible without his encouragement, motivation, inspiration, and guidance.

I would like to thank Pr. Abdallah KHABABA, Pr. Salim CHIKHI, Pr. Mohamed Tarek KHADIR, Dr. Mustapha BOURAHLA, and Dr. Mohamed SAIDI for accepting to be members of the examination committee for this thesis and for taking time to read and review it. Their suggestions will be taken into account and will surely significantly improve the final version.

Last but not least, I wish to thank my family for their love, encouragement and support.

Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER 1

Introduction

Contents

1.1 Overview

1.1.1 Data mining and optimization

Knowledge extraction from data bases, also called data mining, denotes the process of discovering useful, new and understandable information and knowledge from large data bases, data warehouses or others kinds of data repositories . Generally speaking, data mining techniques are classified into two main categories: descriptive techniques and predictive techniques [Fayyad 1996]. In the first category, the aim is to make explicit some information that is present but hidden in data. Methods of this category include: clustering algorithms, association rules, visualization techniques and factorial methods. The second category consists in the extrapolation of new information and predicts unknown information. Among techniques of this type, we find classification algorithms: decision tree, the k-nearest neighbor algorithm and the bayesian classification; estimation algorithms such as neural networks, regression methods in addition to prevision algorithms such as temporal series.

It is worth noticing that the knowledge discovery task performed either by descriptive or predictive techniques may be modeled in some cases as a discrete optimization problem and in other cases as a continuous one. Among data mining

tasks where the nature of information extraction process may be seen as an optimization process, we can cite: association rules, feature selection, clustering and decision trees.

The knowledge generated by association rules is of the form $(A \rightarrow B)$ where A and B present the values of two subsets of features. This knowledge exhibits the links between the values of features in data bases. The process of extracting association rules is often done in two steps: the discovery of the set of frequent itemsets and the extraction of association rules. The first step consists in finding the set of frequent itemsets of size from 1 to k and whose support is greater than or equal to some threshold given by an expert. The problem of finding the set of frequent itemsets may be by modeled as an optimization problem which consists in exploring the search space of 2^n candidate subsets of itemsets where n is the number of items in the data base to find the frequent itemsets [Hipp 2000].

A very similar situation is encountered with feature selection problem. Indeed, feature selection consists in finding the subset of relevant features in a search space of 2^n candidate subsets of features. Similarly, grouping task done in clustering algorithms may be as well modeled as an optimization problem. The question is then how to find the best groups of homogeneous or similar individuals in a population of individuals and what the number of optimal groups is. This problem belongs to the class of NP-complete problems. The number of possible subsets of a set of n individuals is given by the following formula: $B_n = \frac{1}{n}\sum_{k=1}^{\infty}\frac{k^n}{k!}$, where k is the number of classes that may be generated [Dahl 2009].

After this brief survey, it is clear that the nature of some problems of knowledge extraction makes possible to model them as discrete optimization problems. Accordingly, we need fundamental approaches different from classical exact extraction approaches. In this thesis, we have chosen metaheuristic approaches based on collective intelligence of a set of agents to extract knowledge and overcome the complexity difficulty in finding the new information.

1.1.2 Metaheuristics methods for discrete optimization

In this section, we present discrete optimization problems and their resolution methods. More precisely, we are interested in recent metaheuristics based on collective intelligence of a swarm. We present in particular particle swarm optimization, ant colony optimization and firefly algorithm with more detail. Moreover, we present basics of quantum computing and bio-inspired algorithms integrating the concept of this new computer science trend in solving optimization problem.

1.1.2.1 Swarm intelligence methods

Combinatorial optimization is a research domain that is interested in proposing effective methods to find the best possible solutions to a problem in a search space of potential solutions [Martello 2011]. Unfortunately, most of the problems issued from real contexts are intractable and are at least NP-Complete. Thus, solving such problems by exhaustive approaches that examine the whole search space is very expensive (in space and/or in time) and is often impractical as soon as the size of the problem becomes relatively important.

Hence, a number of methods have been proposed in operation research and in artificial intelligence in order to overcome this difficulty. Roughly speaking, these methods can be classified into two main categories [Jourdan 2009]: exact methods and approximate methods. Unlike exact methods that explore the whole search space [Neapolitan 2004], approximate methods explore only a part of the search space to produce good (not necessarily the best) solutions in reasonable time [Eiben 2003].

Among the approximate methods, we find a class of metaheuristics based on the collective swarm intelligence [Krause 2013]. We can cite two popular methods of this class: The first one is the ant colony optimization method proposed by [Dorigo 2005]. It is inspired from the collaboration of a set of ants by using the pheromone (a chemical substance) and that is used to find the optimal path in a graph. This method has been applied to a wide range of discrete optimization problems (see [Dorigo 1999]).

The second method is the particle swarm optimization method (PSO) developed by Kennedy and Eberhart [Kennedy 1995]. To look for the optimal solutions of a problem, this method simulates the flight of a bird's swarm or the movement of insects. Originally, this method has been applied to solve continuous optimization problems. Then, several variants of PSO have been proposed to solve discrete combinatorial optimization problems [Kennedy 1997] including the traveling salesman problem [Hoffmann 2011], the permutation flowshop scheduling problem ([Marinakis 2013]; [Chen 2014]), the data clustering ([Kuo 2011]; [B 2014]) as well as the quadratic assignment problem (QAP) ([Congying 2011]).

During the last years, several other swarm intelligence algorithms have appeared [Yang 2010a]. The functioning principles of these algorithms are inspired from the social behavior of certain living beings such as ants, termites, birds and fishes. Examples of such recent algorithms are: the fish schools algorithm [Neshat 2014], the bat algorithm [Yang 2010c], the cuckoo search algorithm [Yang 2009a], the firefly algorithm ([Yang 2010a], [Yang 2009b]), the bee algorithm

[Karaboga 2014], etc. However, almost all these algorithms have been designed to deal with continuous optimization problems.

The firefly algorithm proposed by ([Yang 2010a], [Yang 2009b]) is based on the light behavior of fireflies. In this approach, each firefly represents a potential solution of the problem so that its brightness is proportional to the value of its objective function. The attractiveness of a firefly depends both on its brightness and its distance from other neighbors. The search space is explored thanks to the dynamics of a collection of fireflies according to the following general rule: the less bright firefly moves towards the brighter one (see [Yang 2010a], [Yang 2009b]). The relative simplicity of the firefly algorithm and its efficiency in exploring the search space have attracted much attention of many researchers who applied it to several optimization problems (see for example [Chatterjee 2012]; [Hassanzadeh 2011]; [Yang 2010b]; [Horng 2012]).

As stated above, the firefly algorithm, as most of the other meta-heuristics, has been proposed originally to deal with continuous optimization problems. Some methods to adapt the firefly algorithm to the discrete context have been proposed, namely, the use of the sigmoid function which transforms the continuous space value into a binary one [Sayadi 2010]; [Falcon 2011]; [Palit 2011]; [Banati 2011]; the definition of a firefly's position in terms of changes of probabilities that will be in one state or the other [Sayadi 2013]; the modification of the movement formula of a firefly [Jati 2011]; the use of the smallest position value which allows the creation of an integer vector of solutions indexed by all the positions of the fireflies' population [Yousif 2011] and the use of the random-key method which translates the firefly position in the continuous space to a value in a combinatory space [Fister Jr 2012].

1.1.2.2 Bioinspired algorithms based on quantum computing

Besides, quantum computing theory provides capacities of parallel treatments and exponential storage of data thanks to the principles of quantum mechanics such as, state superposition, quantum measure, entanglement and quantum gate [Benioff 1980]; [Akama 2015]. Accordingly, quantum computing has become an important source of inspiration in designing efficient tools to solve NP-Complete problems.

Among the most famous quantum algorithms in this domain, we can evoke that of Shor which allows a polynomial time resolution of the well-known NP-Complete problem of number factorization (see [Shor 1994]) and the quantum algorithm of Grover [Grover 1996] which finds a given searched element in a

database in a quadratic time.

Since the early 2000s, a new and promising research area has appeared. It is motivated by the question: How to integrate the quantum computing principles with the optimization meta-heuristic algorithms in order to ensure a good trade-off between their capacities of search space exploration and exploitation, to keep a better diversity of the population throughout the search and to reduce the size of the population (since, as we will see later, each quantum solution will be the superposition of an exponential number of concrete solutions). Some works issued from this research line include: the quantum evolutionary algorithm proposed by [Han 2000], [Han 2002]; the quantum-inspired particle swarm optimization ([Wang 2007c]; [Tazuke 2013]); the quantum-inspired differential evolution ([Pat 2011]; [Hota 2010] and the quantum-inspired tabu search [Chiang 2014].

1.2 Motivation

After the presentation of the nature of different knowledge extraction problems, and the presentation of our problematic below, we need to provide meta-heuristics based on the advantages of metaphors of Swarm intelligence and the advantages offered by the principles of quantum computing for solving the various problems of knowledge extraction. We summarize the motivations of our work in the following points:

- The exploitation of the advantages of the approach of swarm intelligence such as: the parallelism, the decentralization and the cooperation of the swarms of particles to solve complicated combinatorial optimization problems.
- Using the concepts of quantum computation such as the superposition of Q-bits states and quantum observation in the discretization of bio-inspired algorithms originally proposed in the literature for solving continuous optimization problems.
- Taking advantage of the superposition notions of Q-bit states in improving the population's diversity of solutions and reducing the population size of meta-heuristics methods based on the evolution of a population of solutions admissible for combinatorial optimization problem.
- The exploitation of meta-heuristics advantages has been proposed to develop hybrid meta-heuristics and effective in solving the Np-hard optimization problems as presented and shown in my two proposed metaheuris-

tics. The first proposed meta-heuristic is cooperative between the differential evolution and the PSO while the second meta-heuristic is cooperative between the "firefly algorithm" and the PSO.

1.3 Majors contributions

To solve knowledge extraction problems using bio-inspired metaheuristics, our contribution is made in two phases:

- The first phase consists of designing bio-inspired meta-heuristics to solve combinatorial optimization problems in a general manner.
- The second phase is to apply the proposed meta-heuristics to knowledge extraction problems.

In the first phase, we have proposed two bio-inspired algorithms, the first algorithm is called QDEPSO that hybridizes between DE and PSO, and the second algorithm is called QIFAPSO that cooperates between firefly and PSO algorithm. Both proposed algorithms use the concepts of quantum computing to solve combinatorial optimization problems. In the second phase, we applied the QIFAPSO algorithm to solve the problem of selection of attributes.

The first proposed algorithm is called QDEPSO (Quantum inspired Differential Evolution with Particle Swarm Optimization) combines differential evolution (DE), particle swarm optimization method (PSO) and quantum-inspired evolutionary algorithm (QEA) in order to solve the 0-1 optimization problems. In the initialization phase, the QDEPSO uses the concepts of quantum computing to represent and generate the diversity of the initial solutions. The second phase is an alternation between the DE operations (mutation, crossover and selection) and the adapted version of the formula used in the PSO algorithm for the velocity and the position of a particle. The result of this step is to determine the rotation quantum angle to explore the search space of solutions.

The second proposed algorithm is called “Quantum-inspired Firefly Algorithm with Particle Swarm Optimization (QIFAPSO)”. Among other things, it adapts the firefly approach to solve discrete optimization problems. The proposed algorithm uses the basic concepts of quantum computing to ensure a better control of the solutions diversity.

Moreover, we use a discrete representation for fireflies and we propose a variant of the well-known Hamming distance to compute the attractiveness between them. Finally, we combine two strategies that cooperate in exploring the search

space: The first one is the move of less bright fireflies towards the brighter ones and the second strategy is the PSO movement in which a firefly moves by taking into account its best position as well as the best position of its neighborhood. Of course, these two strategies of fireflies' movement are adapted to the quantum representation used in the algorithm for potential solutions.

The third contribution of this thesis is the proposition of an approach for the feature selection problem based on QIFAPSO algorithm. The proposed algorithm QIFAPSO-FS explores the search space constituted of 2^N possible subset of attributes where N stands for the number of input features. The aim is to find the best subset of features thanks to the synergy between the two following strategies: the movement strategy of fireflies as defined in a firefly algorithm and the particle swarm optimization (PSO) strategy.

During the search process, the fireflies change their positions by applying either the FA movement strategy where a less bright firefly moves towards a more brighter firefly or the PSO movement where a firefly changes its position by taking into account its best position up to the current iteration together with the global best position in the whole firefly swarm. This approach uses the concepts of quantum computing, namely qubit superposition in the representation of the probability of selecting features. The evaluation of feature selection is based on the position region concept issued from rough set theory and which allows one to evaluate the relevance of subsets features.

1.4 Thesis organization

Chapter 2 presents the theoretical background of the thesis. It gives a brief review of both bio-inspired algorithms and the feature selection based on such algorithms. It starts with a definition of bio-inspired algorithms and introduces a brief definition for the mains algorithms. It reviews typical related work in feature selection using swarm intelligence techniques and evolutionary computation methods.

Chapter 3 presents the first proposed algorithm QDEPSO for solving knapsack problem. First, it gives a brief introduction to the 0–1 knapsack problem. After that, a detailed description of QDEPSO algorithm is presented. Finally, a summary of an extensive experimental evaluation is given.

Chapter 4 is devoted to the description of the proposed algorithm QIFAPSO. Again, all the relevant issues related to our algorithm are discussed, namely: The binary as well as the quantum representation of fireflies, the initialization of the population, the quantum measure, the method of computing the distance between discrete fireflies, the two movement strategies used (firefly and PSO strategies) and

finally the complete pseudo-code of our algorithm. Finally, it presents and discusses a synthesis of various experimental tests done on the 0–1 knapsack problem both in its one dimensional and its multidimensional variants.

Chapter 5 is devoted to the description of the approach of feature selection based on QIFAPSO algorithm. All the relevant issues related to our algorithm are discussed, namely: the quantum representation for selection feature, the movement strategies, and the evaluation of the feature selection which is based on the region position of the rough set theory and allows one to evaluate the relevance of subsets of subset features.

Chapter 6 summaries the work and attractions overall conclusions of the thesis. Main research ideas and the contributions of the thesis are established as well. It also suggests some possible future research directions.

1.5 Academic publications and communication produced

The following academic papers have been produced as a result of this research.

- Djaafar Zouache and Abdelouahab Moussaoui. Quantum-inspired differential evolution with particle swarm optimization for knapsack problem. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING, vol. 31, no. 5, pages 1757–1773, 2015. http://www.iis.sinica.edu.tw/page/jise/2015/201509_14.pdf. ISSN: 1016-2364, impact factor: 0.33.

- Zouache, D., Nouioua, F., Moussaoui, A. (2015). Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems. Soft Computing journal, 1-19. Springer. DOI: 10.1007/s00500-015-1681-x. ISSN: 1432-7643, impact factor: 1.271.

- Zouache, D. “Genetic algorithms for permutation flow shop scheduling problem”, National Day on applied mathematics,18 December 2013, University of Bordj Bou Arreridj.

- Noufel, D. ,Bensaad, F. and Zouache, D. “ Particle swarm optimization for multiple alignment sequence”, National days of operational research December 12, 2013,University of Bordj Bou Arreridj.

CHAPTER 2

Bio-inspired algorithms for feature selection in classification

Contents

2.1 Introduction

In the classification task, two major problems are encountered. The first one relies on the great number of condition features present in the database to explain the decision feature. This problem may generate high computational complexity in finding the classification model and also, the generated model is often very complex

as it is the case for instance for classification using neural networks where the obtained model contains a huge number of nodes ([Amaldi 1998]; [Cover 1977]). The second problem is the presence of features that are redundant or not sufficiently relevant, which diminishes the quality and performance of the classification.

Feature selection consists in finding the subset of condition features that is the less redundant and the most relevant with respect to the decision feature among all the feature present in the information system. For that purpose, the feature selection task is a crucial task for classification and is done at a preliminary stage. The results of the feature selection task allow one to diminish the complexity of the construction of a classification model and in some cases, to increase the performance of classification.

This chapter gives a brief review of both bio-inspired algorithms and the feature selection based on such algorithms. It initiates with a definition of bio-inspired algorithms and introduces a brief definition for the mains algorithms. It reviews typical related work in feature selection using swarm intelligence techniques and evolutionary computation methods.

2.2 Bio-inspired algorithms

Biologically inspired computing, referred to as Bio-inspired computing, is an area of study that loosely join together subfields linked to the subjects of connectionism, social behavior and emergence. Mainly, it is strongly related to artificial intelligence and can be associated with machine learning as well. Also, Bio-inspired computing depends a lot on computer science, biology and mathematics. To sum it up, it implies using computers to model the living phenomena and studying life for improving the use of computers at the same time. Biologically inspired computing is considered as a main subset of natural computation.

Accordingly, the computational methods inspired by evolution, by nature, and by the brain are being widely employed to solve many complex problems in engineering, computer science, robotics and artificial intelligence. In the following, we present only the bio-inspired algorithms employed in the development and design of our meta-heuristics.

2.2.1 Quantum inspired computation

Quantum computing is a recent field in computer science which is interested in quantum computers using phenomena of quantum mechanics such as state superposition, entanglement and quantum gate [Benioff 1980, Akama 2015]. The fun-

damental information unit in quantum computing is the Q-bit. A Q-bit may be in the state $|0>$, in the state $|1>$ or in a superposition of the states $|0>$ and $|1>$ simultaneously. According to Dirac notation, the Q-bit may be represented as a combination of the states $|0>$ and $|1>$ as follows:

$$|Q>=\alpha|0>+\beta|1> \quad such\ that\ |\alpha|^2+|\beta|^2=1 \tag{2.1}$$

where α and β are complex numbers. $|\alpha|^2$ (resp. $|\beta|^2$) is the probability to find the Q-bit in state 0 (resp. in state 1). A quantum register of size n is then constituted from a set of n Q-bits. It represents a superposition of n Q-bits, i.e., it contains up to 2^n possible values simultaneously. A quantum register is represented by the following notation:

$$\Psi=\sum_{x=0}^{2^n-1} C_X|X> \tag{2.2}$$

The amplitudes C_x satisfy the following propertyÂ :

$$\sum_{x=0}^{2^n-1} |C_X|^2=1 \tag{2.3}$$

The state of a Q-bit can be changed by a quantum gate (Q-gate). A Q-gate is a reversible gate and can be represented as a unitary operator U acting on the Q-bit basis states satisfying $U^+U=UU^+$, where U^+ is the Hermitian adjoint of U. There are several Q-gates, such as the NOT gate, controlled NOT gate, rotation gate, Hadamard gate, etc.[Manju 2014].

2.2.2 Particle swarm optimization method

Particle swarm optimization is a meta-heuristic developed by R. Eberhart and J. Kennedy [Kennedy 1995]. This method is based on the cooperation between particles. Each particle represents a potential solution of the problem. Initially, the particles are put randomly in the search space of the objective function. At each iteration, the particles move by taking into acount their best positions as well as the best positions of their neighbors. The new position of a particle x_i is computed by the following equations:

$$v_i^{t+1}=w.v_i^t+c_1.r_1.(p_i^t-x_i^t)+c_2.r_2.(g^t-x_i^t) \tag{2.4}$$

$$x_i^{t+1}=x_i^t+v_i^{t+1} \tag{2.5}$$

where:

- v_i^t and v_i^{t+1} denote the velocities of the particle i at the iterations t and $t+1$ respectively.
- p_i^t is the best position of the particle i at iteration t.
- g^t is the best position of the neighborhood of particle i at iteration t.
- x_i^t and x_i^{t+1} denote the positions of the particle i at the iterations t and $t+1$ respectively.
- w is the inertia weight.
- c_1 and c_2 are the learning factors.
- r_1 and r_2 are random numbers in the interval $[0,1]$.

The pseudo-code of the PSO algorithm is shown in the algorithm 2.1.

Algorithm 2.1 Particle Swarm Optimization

```
# initialize all particles
Initialize
repeat
  for each particle x_i^t in S do
    # Update the particle's best position
    if f(x_i^t) < f(p_i^t) then
      p_i^t ← x_i^t
    end if
    # Update the global best position
    if f(x_i^t) < f(g^t) then
      g_i^tt
    end if
  end for
  # Update particle's velocity and position
  for each particle x_i^t in S do
    for each dimension d in D do
      v_{i,.d}^{t+1} ← w.v_{i,d}^t + c1.r1.(p_{i,d}^t − x_{i,d}^t) + c2.r2.(g^t − x_{i,d}^t)
      x_{i,.d}^{t+1} ← x_{i,.d}^t + v_{i,.d}^{t+1}
    end for
  end for
until it ≥ MAX_ITERATIONS
```

2.2.3 Differential Evolution

Differential evolution (DE), a stochastic population-based metaheuristic, was originally introduced by Storn and Price [Storn 1997] for solving continuous optimization problems. DE's evolution operator is composed of a differential mutation

aiming at creating a trial vector, which is then used to create one offspring by the crossover operator. Mutation phase sizes can be computed as the weighted differences between randomly selected individual vectors. To create a new population, DE's evolution operators are applied to individuals at each generation. The pseudo-code of the DE is shown in the algorithm 2.2. More precisely, DE's reproduction operators for a minimization problem can be described as follows:

1) **Mutation**
For each solution vector like $X_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{iD}^t)$, $\forall i = 1, \cdots, NP$, in the D-dimensional search space, a mutant vector $X_i^{mt} = (x_{i1}^{mt}, x_{i2}^{mt}, \cdots, x_{iD}^{mt})$ is created according to the following equation:

$$X_{id}^{mt} = X_{r_1 d}^t + F(X_{r_2 d}^t - X_{r_3 d}^t), \forall d = 1, \cdots, D \tag{2.6}$$

with random mutually different indices $r_1, r_2, r_3 \in 1, 2, \ldots, N$, and $F > 0$. The randomly selected integers r_1, r_2 and r_3 are also selected to be different from the running index i, so that NP need to be greater or equal to 4 to allow for this condition. F stands for a real and constant factor that is included in the interval $(0, 2]$ which controls the intensification of differential variations. The superscript t indicates the number of generation. In [Storn 1997], we can see 5 strategies to mutate the X_i^{mt} vector:

- DE/rand/1: $X_i^{mt} = X_{r_1}^t + F(X_{r_2}^t - X_{r_3}^t)$.
- DE/best/1: $X_i^{mt} = X_{best}^t + F(X_{r_1}^t - X_{r_2}^t)$.
- DE/current to best/1: $X_i^{mt} = X_i^t + F(X_{best}^t - X_i^t) + F(X_{r_1}^t - X_{r_2}^t)$.
- DE/best/2: $X_i^{mt} = X_{best}^t + F(X_{r_1}^t - X_{r_2}^t) + F(X_{r_3}^t - X_{r_4}^t)$.
- DE/rand/2: $X_i^{mt} = X_{r_1}^t + F(X_{r_2}^t - X_{r_3}^t) + F(X_{r_4}^t - X_{r_5}^t)$.

Where X_{best}^t represents the individual with the best fitness in the population.

2) **Crossover**
Basically, crossover is introduced in order to augment the diversity among the perturbed mutant vectors. Thus, the trial vector $X_i^{ct} = (x_{i1}^{ct}, x_{i2}^{ct}, \cdots, x_{iD}^{ct})$, $\forall i = 1, \cdots, NP$, is formed as follows: :

$$X_{id}^{ct} = \begin{cases} X_{id}^{mt} & if\ rand() \leq CR\ or\ d = J,\ \forall d = 1, \cdots, D \\ X_{id}^t & otherwise, \end{cases} \tag{2.7}$$

In equation(2.7), $rand()$ is a random number related to d^{th} dimension drawn from a uniform distribution with range $[0, 1]$. The crossover constant in-

cluded in $[0,1]$ is presented by the notation CR. J is a randomly selected index from $1, \cdots, D$.

3) **Selection**

To take the decision whether the trial vector X_i^{ct} should be passed through the next generation (generation $t+1$) or not, it is compared to X_i^t using the following equation:

$$X_i^{t+1} = \begin{cases} X_i^{ct} & if\ f(X_i^{ct}) \leq f(X_i^t) \\ X_i^t & otherwise, \end{cases} \tag{2.8}$$

Algorithm 2.2 Differential Evolution Algorithm

Input: F: differential weight, C: crossover probability, n: population size

```
Initialize the initial population.
while stopping criterion not met do
  for i = 1 to n do
    For each x_i randomly choose 3 distinct vector x_r1, x_r2 and x_r3
    Generate a mutant vector x_i^m using equation 2.6
    Generate a random index J ∈ {1, 2, ..., d}
    Generate a randomly distributed number r_i ∈ [0, 1]
    Obtain X_id^ct by crossover operation using 2.7
    Select and update the solution X_i^{t+1} by equation 2.8
  end for
end while
```

2.2.4 Firefly algorithm

The firefly algorithm has been proposed by X.S. Yang [Yang 2010a, Yang 2009b]. It is based on the light behavior of a population of fireflies. The interaction between fireflies is governed by the following rules [Yang 2010a]:

- All the fireflies are unisex and are attracted by other fireflies independent from their sex.
- The attractiveness of a firefly is proportional to its brightness. The brightness degree of a firefly perceived by another firefly is inversely proportional to the distance between them. In this context, the less bright firefly moves towards the brighter one. If no firefly is brighter than a given firefly, than the later moves randomly.
- The brightness of a firefly is determined by the value of the objective function to optimize.

Thus, the firefly algorithm is based on four main factors [Yang 2010a]:

- **Brightness.** It depends on the objective function. In simple optimization problems, the brightness of a firefly x is reduced to the objective function for $x : I(x) = f(x)$.
- **Attractiveness.** The attractiveness of a firefly is proportional to the brightness perceived by the other neighbors. The attractiveness function may be any function which is monotonically decreasing with respect to the real distance r. A general form of such a function may be:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.9}$$

  where r is the distance between two fireflies, β_0 is the attractiveness for $r = 0$ and γ is a constant (bright absorption coefficient).
- **Distance.** We consider simply the Euclidean distance to measure the distance between two fireflies x_i and x_j: $r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$ where x_{ik} stands for the k^{th} component of the i^{th} firefly.
- **Movement.** The movement of a firefly i attracted by another one j which is brighter is determined by:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha_t \varepsilon_i^t \tag{2.10}$$

  The first and the second terms are due to the attractiveness. The third term is a randomization: α_t is a random parameter which may be constant and ε_i^t is a vector of random real numbers uniformly distributed in $[0, 1]$.

Algorithm 2.3 gives the pseudo-code of the firefly algorithm.

2.3 Feature selection

2.3.1 Definition

Feature selection is usually a search problem for finding an optimal subset of n features out of original N features. Feature selection is essential in some classification problems for rejecting redundant and irrelevant features. It permits reducing system complexity and running time and frequently progresses the classification accuracy [Blum 1997]. The whole search for the best subset of 2^n possible subsets is infeasible for large number of features.

Algorithm 2.3 Firefly Algorithm

Input: β_0, γ

Create an initial population of n fireflies within d-dimensional search: x_{ik}, i = 1, ..., n and k = 1, ..., d;
Evaluate the fitness of the population ($f(x_i)$ is directly proportional to brightness I_i);
while (not termination condition) **do**
  for (i = 1 **:** n: all fireflies) **do**
    for (j = 1 **:** n: all fireflies) **do**
      if ($I_i < I_j$) **then**
        Move firefly i toward j in d-dimension using Eq.2.17;
      end if
      Attractiveness varies with distance r via exp[-r^2];
      Evaluate new solutions and update brightness;
    end for
  end for
end while
Rank the fireflies and find the current best;

2.3.2 Feature selection process

In feature selection algorithm, there are five basic phases [Dash 1997]:

1. The first phase of a feature selection algorithm is the initialization procedure and it is founded on all the original features in the problem.

2. A search procedure to generate candidate feature subsets. It can start with no features, the entire features, or a random subset of features. Several search methods are applied in this feature subset search phase to the exploration for the finest subset of features.

3. An evaluation function to measure the relevance of feature subset.

4. Stopping conditions can be founded on the search procedure or the evaluation function. Conditions based on the search procedure can be whether a predefined number of features are selected and whether a fixed maximum number of iterations have been done.

5. A validation procedure aims to check whether the subset is valid. The validation procedure is not part of the feature selection process itself, but a feature selection algorithm obligation will be validated. The selected feature subset will be validated on the test set.

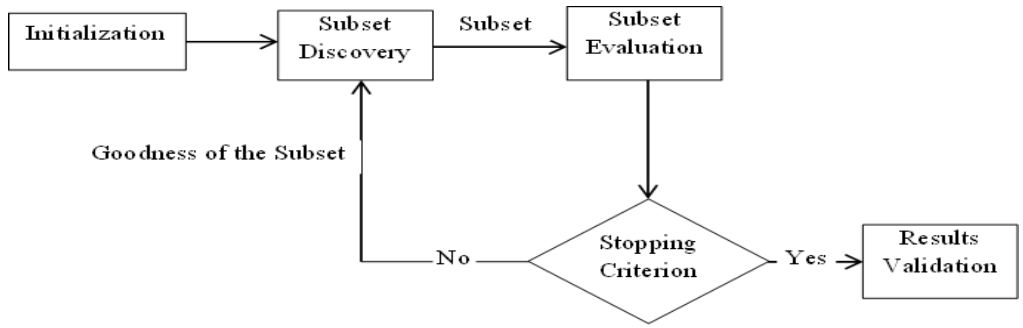


Figure 2.1: Feature selection process [Dash 1997].

2.3.3 Classification of feature selection approaches

Mainly, the existing feature selection methods are grouped into two classes: filter approaches and wrapper approaches [Dash 1997]. A filter feature selection algorithm is independent from any classification algorithm unlike wrappers which use a classification algorithm in the evaluation function. The principle of filter feature selection algorithm is presented in figure 2.2. Figure 2.3 illustrates the principle

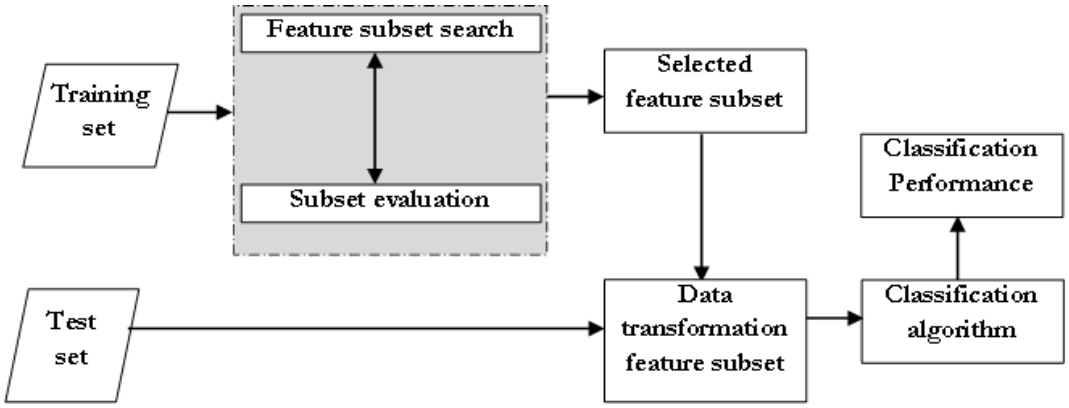


Figure 2.2: A filter feature selection algorithm [Xue 2014].

of wrapper feature selection algorithms. In this second class, the feature selection algorithm occurs as a wrapper about a classification algorithm. To evaluate the quality of feature subsets and guide the exploration, the performance of the classification algorithm is employed in the evaluation function.

2.4 Entropy, mutual information

Information theory offers a means for measuring the information of the random variables. It is related to mathematics and many other fields such as computer sci-

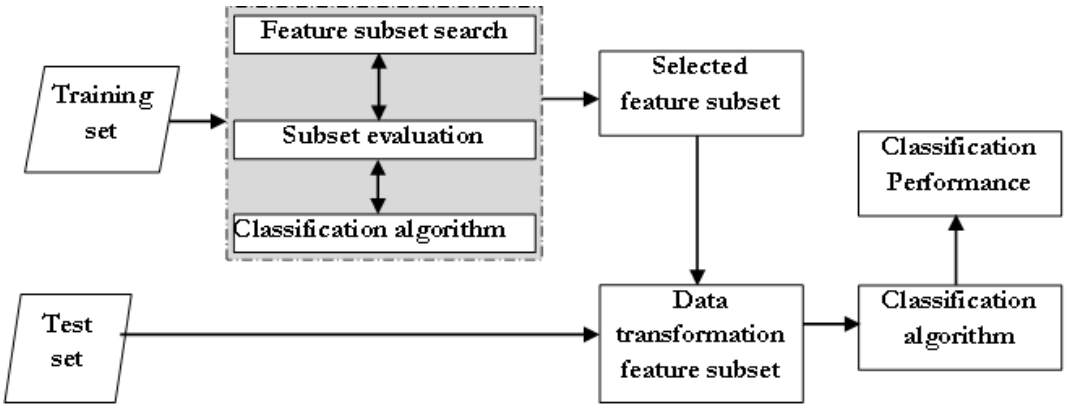


Figure 2.3: A wrapper feature selection algorithm [Xue 2014].

ence, bioinformatics, and electrical manufacturing. Originally, the central grounds of information theory were first introduced by Shannon in 1949 [Shannon 2015]. The entropy is a quantity of the uncertainty of random variables. Let X be a random variable with discrete values, its uncertainty can be measured by entropy $H(X)$, which is given by:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{2.11}$$

where $p(x) = Pr(X = x)$ stands for the probability density function of X. For two discrete random variables X and Y with their probability density function $p(x,y)$, the joint entropy $H(X,Y)$ is defined as:

$$H(X,Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 p(x,y) \tag{2.12}$$

In the case when a certain variable is identified and others are unidentified, the residual uncertainty is measured by the conditional entropy. Shoulder that variable Y is given; the conditional entropy $H(X|Y)$ of X with respect to Y is presented by equation:

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 p(x|y) \tag{2.13}$$

In the equation above, $p(x|y)$ presents the posterior probability of X given. Thus, If X totally depends on Y, $H(X|Y)$ is zero. This implies that no additional information is needed to define X when Y is already identified. Moreover, $H(X|Y) = H(X)$ means that knowing Y has nothing to do with the probability of observing X. The common information between two random variables is defined as mutual information [Cover 2012]. Given variable X, how abundant information

one can gain around variable Y , which is mutual information $I(X;Y)$.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= -\sum_{x\in X, y\in Y} p(x,y)\log_2 \frac{p(x,y)}{p(x)p(y)} \end{aligned} \tag{2.14}$$

In the equation above, if two variables X and Y are strictly related, then the mutual information $I(X;Y)$ is high. Yet, $I(X;Y) = 0$ holds in the case where X and Y are completely independent. Filter feature selection has widely employed information theory, especially mutual information for measuring the link between the selected features and the class variable.

2.5 Basic notions on Rough set theory

Rough set theory is a mathematical approach proposed by Zdzislaw Pawlak [Pawlak 1982] to analyze uncertain, ambiguous and imprecise data. In this section, we introduce some concepts of rough set theory in the context of incomplete information systems that will be used in this chapter. These concepts are: information system, indiscernibility, set approximation, neighborhood rough sets and features dependency.

2.5.0.1 Information system

An information system is defined by the tuple $(U, A = C \cup D, V, fa)$, where U is a finite non empty set of objects, C is a non-empty set of features called the set of condition features, D is a non-empty set of features called the set of decision features and $C \cap D = \phi$; $V = \cup_{a\in A} V_a$, with V_a is the set of values or the feature domain $a \in A$; $f_a : U \to V_a$ is an information function defined from U towards V_a .

2.5.0.2 Indiscernibility

For every subset of condition features $B \subset C$, there is an associated equivalence relation $IND(B) = \{(x,y) \in U^2 | \forall a \in B, f_a(x) = f_a(y)\ or\ f_a(x) = *or\ f_a(y) = *\}$, where $IND(B)$ is called B-indiscernibility relation [Pawlak 2007]. This relation means that couples of objects $(x,y) \in U^2$ are indiscernible by the set of features B. However, this definition is valid just for incomplete information system containing a set of qualitative features. However, in real applications, the information systems are generally heterogeneous, i.e., contain a mixture of qualitative continuous features.

In this case, the indiscernibility of a subset of condition features $B \subset C$ and B=$B_1 \cup B_2$ where B_1 is a subset of qualitative features and B_2 is a subset of continuous features, is given by the following formula:

$$IND(B) = \{(x,y) \in U^2 | \forall a \in B_1, f_a(x) = f_a(y)\} \cup \{(x,y) \in U^2 | \forall a \in B_2, |f_a(x) - f_a(y)| \leq \varepsilon_a\} \tag{2.15}$$

where ε_a is a non negative real number corresponding to the feature $a \in B2$. The relation $IND(B)$ generates a partition $U/IND(B) = \{[x]_B | x \in U\}$ over U, where $[x]_B$ denotes the equivalence classes.

2.5.0.3 Object neighborhood

The neighborhood of an object x is the maximal set of objects that are possibly indiscernible by the subset of features B with the object x.

$$S(x)_B = \{y | (x,y) \in IND(B), y \in U\} \tag{2.16}$$

2.5.0.4 Set approximation

Let $B \subset C$ be the set of condition features, $S(x)_B$ be the neighborhood of each object $x \in U$ by the subset of features B. The approximation of the set of object $X \subset U$ by using the neighborhood $S(x)_B$ is given by the lower $\underline{B}X$ and the upper approximation $\overline{B}X$. The lower approximation of X is defined by:

$$\underline{B}X = \{x \in U | S(x)_B \subseteq X\} \tag{2.17}$$

The upper approximation of X is defined by:

$$\overline{B}X = \{x \in U | S(x)_B \cap X \neq \phi\} \tag{2.18}$$

The positive region of X is represented by the lower approximation and denoted by $POS_B(X)$.

2.5.0.5 Dependency of Attributes

Let $I = (U, A = C \cup D, V, f_a)$ be an incomplete information system. The partitioning of the universe U by the indiscernibility relation of the decision feature D is: $U/IND(D) = \{D_1, D_2, \cdots, D_k\}$ with $U = \cup_{i \in \{1,\cdots,k\}} D_i$, $\underline{B}D_i$ is the lower approximation of each partition D_i by the set of condition features B. The positive region of the decision feature D which respects the set of set of condition features B, denoted by

$POS_B(D)$ is given by:

$$POS_B(D) = \cup_{D_i \in U/IND(D)} POS_B(D_i) \tag{2.19}$$

with $POS_B(D_i) = \underline{B}D_i = \{x \in U | S(x)_B \subseteq D_i\}$ The dependency degree between the set B of condition features and the set decision feature is given by the following formula:

$$\gamma_R(D) = \frac{POS_B(D)}{|U|} \tag{2.20}$$

2.6 Background of approaches for feature selection

2.6.1 Mutual information based approach

The mutual information is an efficient tool in evaluating the relevance and redundancy between features. The MIFS feature selector algorithm, developed by battiti [Battiti 1994], used such theory to join between inputs features and outputs features for the problems of classification. MIFS selects the feature that maximizes the information of the class feature, and corrected by subtracting a measure proportion to the average MI with the previously selected features. The method MIFS-U [Kwak 2002], is introduced by Kwak and Choi for increasing MIFS in solving nonlinear problems, which in general, makes a better estimation of the MI between inputs attributes and outputs classes than MIFS. Another variant of MIFS is the min-redundancy max-relevance [Peng 2005], MRMR for short. The latter presents two-steps for feature selection algorithm. In the first step, the mRMR uses incremental selection method to find a candidate feature set. In the second step, the backward and forward selection is used to find a compact feature subset from the candidate feature set.

However, the major drawback of such methods is that they selects one feature with maximum criterion in each pass, without taking into account the interaction between groups of features. Thus, numerous methods have been proposed in this scope to avoid this inconvenience, among which we cite: Conditional Likelihood Maximisation [Brown 2012] , Low bias histogram-based estimation of mutual information [Hacine-Gharbi 2012] and Joint Mutual Information Maximisation Feature selector [Hacine-Gharbi 2012].

2.6.2 Rough set based approaches

In general, rough set methods for feature selection are classified into two groups; greedy methods and stochastic methods. The first type usually employs rough set

attribute significance as heuristic information. It starts off with an empty set or attribute core and then adopts forward selection or backward elimination. In forward selection, it begins with an empty set and adds features. Backward elimination is the reverse; it begins with a full set and deletes features incrementally. Hu developed a reduction algorithm employing the positive region-based attribute significance just as the guiding heuristic knowledge ([He 2006], [Wang 2007a]). Using conditional entropy-based attribute significance, Wang proposed a conditional information entropy based reduction algorithm [Wang 2004]. Approximate reducts and approximate entropy reducts were also the focus of research studies. Hu and other researchers proposed reduction algorithms using the discernibility matrices-based attribute significance as the guiding heuristic information ([Wang 2001],[Hu 2003], [Li 2010]). The positive region and conditional entropy-based methods choose a minimal feature subset that fully describes all concepts in a given dataset. The discernibility matrix-based method is to select a feature subset with great discriminatory power.

2.6.3 Metaheuristics approaches based on rough set for feature selection

In the feature selection problem, two major issues are essential. The first one is how to explore the search space of the candidate subsets of features to search for the optimal subset of features. How to evaluate the relevance of a given candidate subset of features is the second issue. According to the exploration strategy and the evaluation criterion, feature selection methods are grouped into two categories; the filter and the wrappers approaches [Langley 1994].

Mainly, these two kinds of approaches differ in the use of classification algorithm by the wrappers approaches to evaluate the relevance of a subset of features during the feature selection process. In contrast, the search of subsets of features and the evaluation of their relevance, in filter approaches, are done independently from the classification algorithm. On the whole, wrapper approaches perform better than filter approaches in terms of quality. Yet, they are more expensive in terms of computations [Dash 1997]. A variety of decision theories dealt with evaluating the relevance of a feature or a subset of features in filter approaches. Among these approaches, we can cite: distance measures [Kwak 2002], dependency measures [Yu 2004], consistency measures [Yuan 1999], mutual information [Estévez 2009], rough set [Pawlak 1982], fuzzy rough set [Wygralak 1989] and Dempster Shafer theory [Dempster 1967].

It should be noted that feature selection is a difficult task to solve due to the difficulty of exhaustively exploring the search space whose size is 2^n combina-

tions for n decision features. It is worth noticing also that the results of the search strategy impact explicitly the quality of solutions and the complexity of their computation. In the literature, several search strategies exist such as greedy selection [Caruana 1994], backward selection [Marill 1963] and metaheuristics approaches.

Rough set theory, a mathematical approach proposed by Pawlak [Pawlak 1982], is viewed as an effective technique for feature selection, extraction of association rules and knowledge discovery from categorical data ([Degang 2007], [Pawlak 2007], [Slowinski 2000], [Swiniarski 2003]). Several metaheuristic approaches have been proposed to solve the feature selection problem and they have clearly proven to be more efficient in exploring the search space, i.e., finding the global solution while avoiding the premature convergence problem.

The most popular metaheuristics in this domain are: particle swarm optimization (PSO) [Kennedy 1995], ant colony optimization (ACO) [Dorigo 2005] as well as genetic algorithms [Holland 1992]. A great number of works existing in the literature are concerned with the hybridization of metaheuristics approaches and rough set theory. To address feature selection problems, Jensen and Shen [Jensen 2005] apply ACO in order to find a small reduct in rough set theory. Afterward, Ke et al. [Ke 2008] and Chen et al.[Chen 2010] also successfully use ACO and rough set theory to solve feature selection problems. However, the datasets used in these papers have a relatively small number of features (the maximum number is 70), Wang et al.[Wang 2007b] propose a filter feature selection algorithm based on an improved binary PSO and rough set, and fnding rough set reducts with fish swarm algorithm [Chen 2015].

2.7 Chapter summary

This chapter reviewed the theoretical background of the thesis. It provides a brief review of both bio-inspired algorithms and the feature selection based on such algorithms. It starts with a definition of bio-inspired algorithms and introduces a brief definition for mains algorithms. This chapter reviewed the main concepts of feature selection, process of feature selection, classification of feature selection methods, entropy and mutual information, rough set theory. Finally, this chapter also reviewed the related work about the approaches for feature selection.

CHAPTER 3

QDEPSO for Knapsack Problem

Contents

3.1 Introduction

Evolutionary algorithms represent a very efficient meta-heuristics that is widely used for global optimization problems. However, these meta-heuristics demand large memory to represent the population of solutions as well as large computational time in order to find the optimal solution. The emergence of quantum computing [Benioff 1980, Feynman 1982] , offers treatment capacities and exponential storage thanks to the principles of quantum theory such as superposition of quantum states, entanglement and quantum interference. Several researchers studied the effect of introducing inspired operations by quantum computing in

evolutionary algorithms to maintain a good balance of exploration and exploitation ([Han 2000], [Han 2002], [Talbi 2004], [Chou 2011], [Chang 2010]).

Han and Kim were the first to propose a new quantum inspired evolutionary algorithm (QEA) combining classical evolutionary algorithm with the quantum computing's concepts such as the quantum bit and the quantum rotation gate [Han 2000], while others modified QEA algorithm to enhance its performance. The addition of the mutation operation [Zhou 2006], the crossover operation [Xiao 2008], the new termination criterion and the new rotation gate [Han 2004] were introduced in order to avoid the problem of premature convergence.

Particle swarm optimization (PSO) algorithm proposed by Kennedy and Eberhart is a stochastic optimization method consisting of a candidate solutions' population known as particles [Kennedy 1995]. These particles are displaced in the search space to find an optimal solution. The movement of each particle is guided by its best known position (pbest) and neighbor best position (gbest). Motivated by the success of QEA algorithm and the excellent ability of PSO in global optimization, many algorithms have introduced the quantum computing in PSO in order to guarantee the global convergence. Wang et al presented a novel quantum swarm evolutionary algorithm (QSE) which uses update formula of velocity and position to update quantum solutions [Wang 2007c]. Further, to increase the diversity of the population's solutions and to improve the exploration of the search space by PSO, the mutation [Liu 2006], the artificial immune [Dong 2009], the crossover [Lin 2010] operators and modified formulae of PSO [Wang 2011] were introduced.

The differential evolution (DE), introduced by Storn and Price [Storn 1997], is based on vector solutions' population for global optimization. The DE algorithm uses simple operations which include mutation, crossover and selection to explore the search space. As will follow closely, there are several algorithms which hybridize DE algorithm with quantum inspired to increase the global search ability of DE. Hota and Pat [Hota 2010] proposed an adaptive quantum-inspired differential evolution algorithm for 0-1 Knapsack problem (AQDE) which uses the quantum representation, the measurement introduced by QEA algorithm for adapting operations of mutation, the crossover and the selection operators of DE to quantum individuals. Further, the quantum interference operator [Draa 2011] and the mutation operation of genetic algorithm [Su 2008] were introduced in QDE too.

This chapter proposes a new hybrid algorithm combining the differential evolution algorithm (DE), the particle swarm optimization method (PSO) and quantum evolutionary algorithm (QEA) to solve the knapsack 0-1 problem. In this algorithm, we introduce the concepts of quantum representation, quantum measurement and rotation angle to update quantum individuals. We adapt the operations

of DE (mutation, crossover, selection) to quantum individuals. Finally, we use PSO formula to determine the quantum angle rotation.

The rest of the chapter is organized as follows: A brief introduction to the 0-1 knapsack problem is presented in section 3.2. The proposed QDEPSO algorithm is described in section 3.3. Section 3.4 summarizes extensive experimental evaluation.

3.2 Knapsack Problem

The 0-1 knapsack problem is a classical combinatorial optimization problem. It is studied in several fields like decision making, complexity theory, cryptography, budget controlling; etc. This problem was demonstrated to be NP-hard [Pisinger 2005]. The Knapsack 0-1 problem may be formulated as follows:

- Given a set of m items: $X = (x_1, x_2, x_3, ..., x_m)$.
- Each item x_i has a weight wi and a profit p_i.

The problem is to select a subset from set of m items without exceeding a given weight capacity C where the overall profit is maximized.

We can formulate mathematically the problem as follows:

$$Maximize \sum_{i=1}^{n} p_i x_i \tag{3.1}$$

Subject to

$$\sum_{i=1}^{n} a_i x_i \leq C,\ x_i \in \{0,1\}\ where\ 1 \leq i \leq n \tag{3.2}$$

With x_i: can take either the value 1 (selected) or the value 0 (not selected).

3.3 The proposed algorithm

In the following, we present the proposed algorithm QDEPSO which hybrids differential evolution, particle swarm optimization and quantum evolutionary algorithm for the knapsack problem. The QDEPSO architecture contains three essential modules. The first module includes the generation of quantum individual, the observation operator and the objective function. The second module is composed of two main operations of DE (mutation, crossover) and quantum observation. Finally, the third module contains the selection operator of DE.

The selection operator selects the better one among the current individual and the trial individual created by the mutation and crossover operations. In the case where the trial individual is worst performing compared to individual of the current population, the QDEPSO injects the formula of PSO and the interference of QEA to maintain the diversity of the population of solutions as well as to accelerate the search for the global optimal solution. Figure 3.1 summarizes the global architecture of the QDEPSO algorithm.

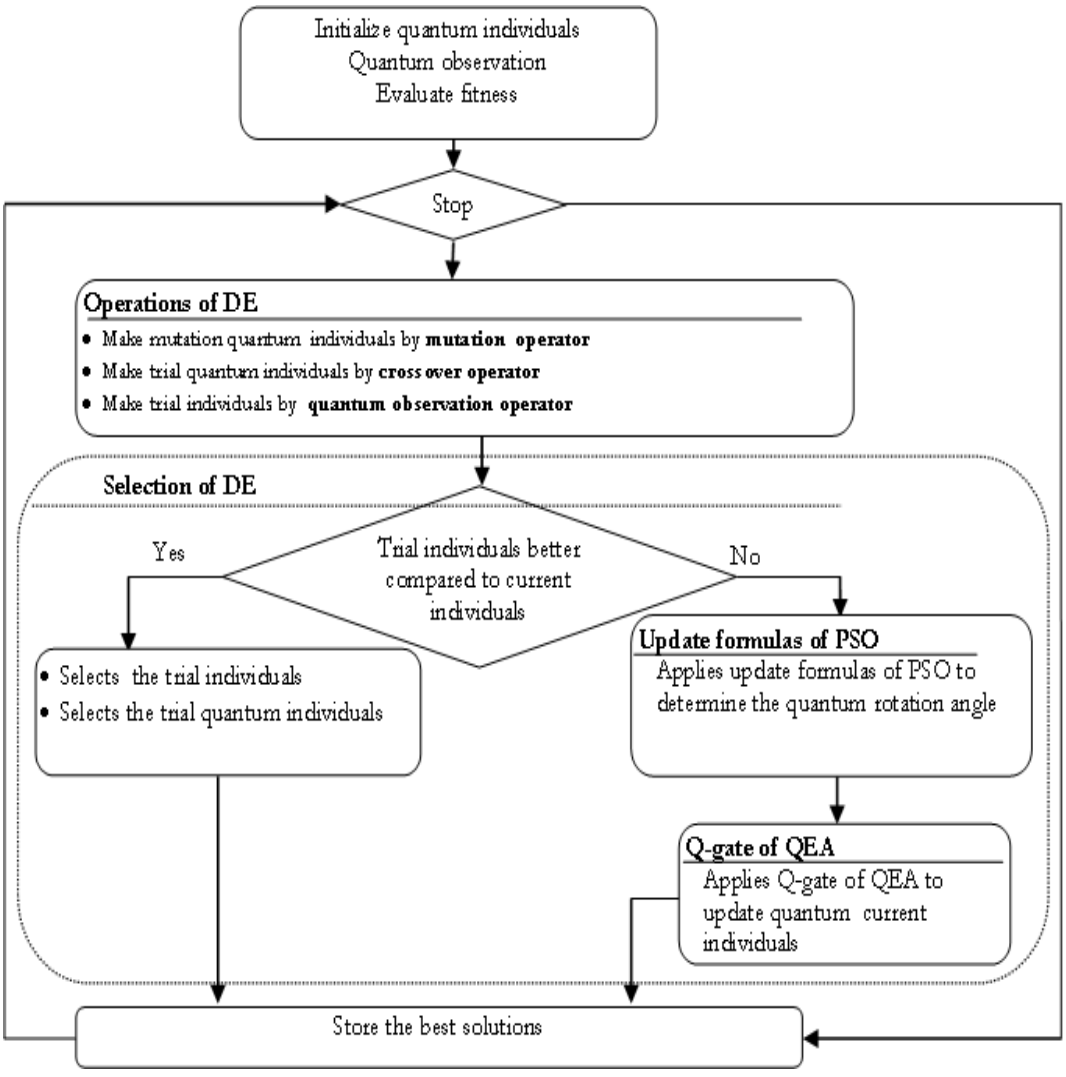


Figure 3.1: The global architecture of the QDEPSO algorithm.

3.3.1 Binary representation of items selection

The choice of a representation for individuals is a very important and a crucial issue in evolutionary algorithms. The QDEPSO which uses the binary coding is the most suitable for the items' selection. Each individual X is represented as a vector of length m (m is the individual size): $X = (x_1, x_2, \ldots, x_m)$, where n is the number of items.

$$\begin{cases} x_i = 1 \; if \; the \; item \; x_i \; is \; selected \\ x_i = 0 \; if \; the \; item \; x_i \; is \; rejected \end{cases}$$

The following example illustrates the binary representation of the items' selection x_1 and x_4 from the items set V: $V = (x_1, x_2, x_3, x_4) \rightarrow X = (1, 0, 0, 1)$.

$P(t) = \{X_1^t, X_2^t, \ldots, X_n^t\}$ is the individual population based on classical bit at the t^th generation, where n is the size of population; m is the length of individual X_j^t.

3.3.2 Quantum representation

We adopt the representation of AQDE [Hota 2010], where each quantum individual q corresponds to a vector q_θ.

q_θ:is a string of variables $\theta_i (1 \leq i \leq m)$,with $\theta_i \in [0, 2\pi]$

$$q_\theta = (\theta_1, \theta_2, \cdots, \theta_m) \tag{3.3}$$

Each quantum individual q is a string of quantum bits:

$$q = \begin{bmatrix} cos(\theta_1) & |cos(\theta_2) & | & \ldots & | & cos(\theta_m) \\ sin(\theta_1) & |sin(\theta_2) & | & \ldots & | & sin(\theta_m) \end{bmatrix} \tag{3.4}$$

The probability amplitude of one quantum bit is defined with a pair of numbers $(\cos(\theta_i), \sin(\theta_i))$. $|\cos(\theta_i)|^2$ represents the probability of rejecting item x_i and $|\sin(\theta_i)|^2$ represents the probability of selecting item x_i. Of course $\cos(\theta_i)$ and $\sin(\theta_i)$ satisfy (3.4):

$$|\cos(\theta_i)|^2 + |\sin(\theta_i)|^2 = 1 \tag{3.5}$$

$Q(t) = \{q_1^t, q_2^t, \cdots, q_n^t\}$: is the quantum population at the t^th generation, where n is the size of the population, and m is the length of the qubit quantum individual.

3.3.3 Initialization

In the initialization step, a quantum individual population represents the linear superposition of all possible states with equal probability. In the case of knapsack

problem, a quantum individual represents a superposition of all possible combinations of selected items. For this, each vector $q_{\theta i}$ is initialized by:

$$q_{\theta i}^{0} = ((\frac{\pi}{4}).r_{i1}, (\frac{\pi}{4}).r_{i2}, \cdots, (\frac{\pi}{4}).r_{im}) \tag{3.6}$$

Where r_{ij}: is an odd integer generated randomly in the set $r_{ij} \in \{1,3,5,7\}$ corresponding to $\theta \in \{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\}$. The qubit q_{ij} corresponding to θ_{ij} is probably located in one of the four quadrants of the unit circle as described in Figure 3.2. This initialization provides higher diversity in initial population of quantum individuals.

$$q_i^0 = \begin{bmatrix} cos(\theta_{i1}^0) & |cos(\theta_{i2}^0) & | & \dots & | & cos(\theta_{im}^0) \\ sin(\theta_{i1}^0) & |sin(\theta_{i2}^0) & | & \dots & | & sin(\theta_{im}^0) \end{bmatrix} \tag{3.7}$$

With: $|\cos(\theta_{ij}^0)|^2 = \frac{1}{2}$ *and* $|sin(\theta_{ij}^0)|^2 = \frac{1}{2}$

3.3.4 Quantum observation

In this operation, quantum individuals are transformed by the projection qubits into binary individuals. Each qubit transforms a binary bit. In the knapsack problem, we propose to make the observation and the reparation of individuals simultaneously (see the algorithm 3.1).

Algorithm 3.1 Observe and repair (X: binary individual) from (q: quantum individual)

1: $x_{j\in\{1,\cdots,m\}} \leftarrow 0$; #Initializes the bits of individual X to zeros
2: $totalw \leftarrow 0$; # the weights' total of the individual X is initialized to zero
3: **while** $(totalw \leq C)$ **do**
4: $j \leftarrow rand[1,m]$; #generation of the random integer $j \in \{1,\cdots,m\}$
5: **if** $(x_j = 0)$ **then**
6: $r \leftarrow random(0,1)$; #add the weight a_j of the selected item to the total weights $totalw$
7: **if** $(r > |cos(\theta_j)|^2)$ **then**
8: $x_j \leftarrow 1$
9: $totalw \leftarrow totalw + a_j$
10: **end if**
11: **end if** #the end of the loop, the total weight exceeds capacity C. For this, we extract the item x_j from the selected items list
12: **end while**
13: $x_j \leftarrow 0$;
14: $totalw \leftarrow totalw - w_j$;

3.3.5 Mutation operation

The mutation operation is used to differentiate the population of solutions. There are two important factors in the mutation of solutions vectors. The first one is the random selection of parent's vectors to explore the search space and the second factor is the coefficient F of controling differentiation. In the algorithm QDEPSO, we adapt the strategy of selecting parent vectors proposed by K. Price and R. Storn (Scheme DE/best/2) [Chang 2010] . In this strategy, the mutation vector $q_{\theta i}$ is generated by making the difference between two target vectors $q_{\theta r1}$ and $q_{\theta r2}$, randomly selected. Then, this difference is weighted by a factor of differentiation's control $F \in [0,2]$. These differences are then added to a third vector $q_{\theta best}$ which corresponds to the current best individual X_{best}.

$$q_{\theta i}^{mt} = q_{\theta best} + F.(q_{\theta r1} - q_{\theta r2}) \tag{3.8}$$

q_i^{mt} is the mutation quantum individual which corresponds to the mutant vector $q_{\theta i}^{mt}$, it is given by:

$$q_i^{mt} = \begin{bmatrix} cos(\theta_{i1}^{mt}) & |cos(\theta_{i2}^{mt}) & | & \dots & | & cos(\theta_{im}^{mt}) \\ sin(\theta_{i1}^{mt}) & |sin(\theta_{i2}^{mt}) & | & \dots & | & sin(\theta_{im}^{mt}) \end{bmatrix} \tag{3.9}$$

3.3.6 Crossover operation

The trial vector $q_{\theta ij}^{ct}$ is generated by crossover between the target vector $q_{\theta ij}^{t}$ and the mutant vector $q_{\theta ij}^{mt}$. The formula of crossover operation is presented as follows:

$$q_{\theta ij}^{ct} = \begin{cases} q_{\theta ij}^{mt} \; if \; (rand_j(0,1) \leq CR^t) \; or \; (j = I_{rand(1,m)}) \\ q_{\theta ij}^{t} \; if \; (rand_j(0,1) > CR^t) \; or \; (j \neq I_{rand(1,m)}) \end{cases} \tag{3.10}$$

with $CR \in [0,1]$ is randomly generated in each generation t. The trial quantum individual q_i^{ct} which corresponds to the trial vector $q_{\theta i}^{ct}$ is given by:

$$q_i^{ct} = \begin{bmatrix} cos(\theta_{i1}^{ct}) & |cos(\theta_{i2}^{ct}) & | & \dots & | & cos(\theta_{im}^{ct}) \\ sin(\theta_{i1}^{ct}) & |sin(\theta_{i2}^{ct}) & | & \dots & | & sin(\theta_{im}^{ct}) \end{bmatrix} \tag{3.11}$$

3.3.7 Selection operation

The result of the observation of the population of trial quantum individuals q_{ct} gives a population of trial individuals $P^{ct} = \{X_1^{ct}, X_2^{ct}, \cdots, X_m^{ct}\}$. The selection operation operates on the current individuals and the trial individuals respectively in

the following manner:

$$X_i^{t+1} = \begin{cases} X_i^{ct} \ if \ (f(X_i^{ct}) > f(X_i^t)) \\ X_i^t \ otherwise \end{cases} \tag{3.12}$$

$$q_i^{t+1} = \begin{cases} q_{\theta i}^{ct} \ if \ (f(X_i^{ct}) > f(X_i^t)) \\ update \ q_{\theta i}^t \ by \ formulae \ of \ PSO \ and \ AQE \ , \ otherwisee \end{cases} \tag{3.13}$$

3.3.8 Quantum rotation gate

A quantum rotation gate $U(\Delta\theta_i)$ is used to update the values of the Q-bits of each quantum individual as follows [Feynman 1982] :

$$U(\Delta\theta_i) = \begin{pmatrix} cos(\Delta\theta_i) & -sin(\Delta\theta_i) \\ sin(\Delta\theta_i) & cos(\Delta\theta_i) \end{pmatrix} \tag{3.14}$$

Where, $\Delta\theta_i$ is a rotation angle of each qubit to the state 0 or 1 depending on its sign.

3.3.9 Adaptation of the PSO formula

In order to determine the rotation angle of each qubit, we adapt the formulae of updating the velocity and position of PSO which are expressed as Eqs. **??** and **??**, respectively :

$$\Delta\theta_{ij}^{t+1} = \Delta\theta_{ij}^t + c_1 r_1 \Delta\theta_{(x_{ij},b_{ij})}^t + c_2 r_2 \Delta\theta_{(x_{ij},bg)}^t \tag{3.15}$$

$$q_{\theta ij}^{t+1} = q_{\theta ij}^t + \Delta\theta_{ij}^{t+1} \tag{3.16}$$

- $\Delta\theta_{ij}^t$: is the rotation angle of the previous generation t, with $\Delta\theta_{ij}^0$ initialized to 0.
- $\Delta\theta_{(x_{ij},b_{ij})}^t$: is determined by the qubit j of quantum individual q_i, the bit j of individual x_i and the bit j of the local optimal solution b_i, as shown in table 3.1.
- $\Delta\theta_{(x_{ij},bg)}^t$: is determined by the qubit j of quantum individual q_i, the bit j of individual x_i and the bit j of global optimal solution bg, as illustrated in table 3.1 .
- r_1 and r_2 are random numbers in the interval [0, 1].
- c_1 and c_2 are the learning factors.

Table 3.1: Lookup table of the rotation angle (x: binary individual, b: best local solution, bg: best global solution, $f(.)$: fitness function)

x_j	b_j	bg_j	$f(x) > f(b)$	$f(x) > f(bg)$	$\Delta\theta_{(x_j,b_j)}$	$\Delta\theta_{(x_j,bg_j)}$
0	1	1	false	false	0.01π	0.01π
1	0	0	false	false	-0.01π	-0.01π
0	1	1	false	true	0.01π	0
1	0	0	false	true	-0.01π	0
0	1	1	true	false	0	0.01π
1	0	0	true	false	0	-0.01π
0	1	0	false	true or false	0.01π	0
1	0	1	false	true or false	-0.01π	0
0	0	1	true or false	false	0	0.01π
1	1	0	true or false	false	0	-0.01π

3.3.10 Outlines of QDEPSO algorithm

The algorithm 3.2 summarizes how the Quantum-Inspired Differential Evolution Particle Swarm Optimization (QDEPSO) algorithm works.

Algorithm 3.2 Pseudo code of QDEPSO

$t \leftarrow 0$
Initialize $Q(0)$ by equations 3.6 and 3.7.
Make and repair $P(0)$ from $Q(0)$
Evaluate fitness of $P(0)$
while $(t < Max\_generation)$ **do**
  $t \leftarrow t + 1$
  Make and repair $P(t)$ from $Q(t-1)$
  Apply mutation on $Q(t)$ by equations 3.8 and 3.9.
  Obtain $Q'(t)$ by crossover operation using equations 3.10 and 3.11.
  Make and repair $P'(t)$ from $Q'(t)$
  Evaluate fitness of $P'(t)$
  if selection criterion met **then**
    Update $P(t+1)$ and $Q(t+1)$ by equations 3.12 and 3.13.
  else
    Update $Q(t+1)$ using by formulae of PSO and QEA (equations 3.14-3.16).
  end if
end while

3.4 Experimental results

In order to evaluate the performance of the proposed QDEPSO algorithm, we make two experiments. All computational experiments are conducted using Matlab7. In the first experiment of 0-1 knapsack problem, weights a_i and respective profits p_i were calculated as follows [Han 2002] :

- $a_i = rand[1,10]$ generates an integer in $1,2,...,10$ uniformly at random.
- $p_i = a_i + 5$ and the knapsack capacity C used is : $C = \frac{1}{2}\sum_{i=1}^{m} a_i x_i$
- We have used various size items of N ranging from 50 to 3000. And the maximum number of generations in all cases was chosen as 1000.

The proposed algorithm QDEPSO is compared with the quantum inspired algorithms to solve the knapsack problem named: QEA [Han 2002] , AQDE [Hota 2010] and QSE [Wang 2007c] to present the hybridization utility of the three algorithms (DE, QEA and PSO) in the global optimization. The used parameters for the preceding algorithms are presented in the following table where the population size is 10. The average, the best and the worst profits, as well as, the respective standard in deviations of 30 independent runs are presented in table 3.3 , figures 3.2-3.4 and 3.5-3.7.

Table 3.2: The parameters of algorithm.

	QEA	QSE	AQDE	QDEPSO
Quantum rotation angle	0.01π	/	/	0.01π
Parameters of PSO	/	w=0.7298, c1= 1.42, c2= 1.57	/	w=0.9, c1= c2= 1
Differentiation Control (F)	/	/	$random(0,1) \times random(0,1) \times 0.1$	$random(0,1) \times random(0,1) \times 0.1$
Crossover control (CR)	/	/	rand('norm',0.5, 0.0375): generates a random number from the gaussian distribution.	rand('norm',0.5, 0.0375): generates a random number from the gaussian distribution.

In order to provide the performance of the injection of formulae of PSO and Q-gate of QEA in the selection operation of DE, we have compared the QDEPSO algorithm in the three cases; with the DE selection, without the DE selection, and with the roulette wheel selection of genetic algorithm (GA). Table 3.4 and figures 3.8-3.9 show the results of the QDEPSO comparison.

In order to present the utility of the injection of PSO formulae and Q-gate of QEA in the selection operation of DE, we have compared the QDEPSO algorithm in the three cases; with the DE selection, without the DE selection, and with the roulette wheel selection of genetic algorithm (GA). The results of the comparison are presented in table 3.4 and figures 3.8- 3.9.

In order to show the importance of the crossover operation of the DE in exploring the search space, we have performed a comparison between the QDEPSO

Table 3.3: Experimental results of the 0-1 knapsack problem (experiment 1).

Item size		QEA	QSE	AQDE	QDEPSO
50	Best	302	297	292	302
	Average	300.63	294.26	287.2	302
	Worst	297	290	282	302
	Std	2.23	2.41	2.97	0
250	Best	1517	1446	1417	1552
	Average	1502.9	1427.7	1397.6	1548.5
	Worst	1496	1412	1382	1542
	Std	4.4562	8.204	7.3178	2.6749
500	Best	2946	2799	2772	3078
	Average	2917.3	2783	2732	3068.2
	Worst	2907	2763	2717	3058
	Std	8.8198	9.2364	11.3304	4.8215
1000	Best	5695	5460	5382	6091
	Average	5662.5	5442.2	5364.4	6071.9
	Worst	5633	5422	5342	6048
	Std	12.7028	11.2975	10.1018	10.3636
1500	Best	8464	8128	8198	9079
	Average	8439.4	8082.8	8178.7	9032
	Worst	8414	8039	8149	8990
	Std	15.1535	20.6722	13.6188	20.5552
2000	Best	11217	10813	10951	12020
	Average	11191	10781	10900	11980
	Worst	11164	10747	10865	11931
	Std	14.7202	16.2827	24.6167	23.348
2500	Best	13907	13466	13569	14910
	Average	13866	13394	13523	14837
	Worst	13839	13342	13482	14751
	Std	19.0504	24.5971	23.5438	34.7997
3000	Best	16604	16071	16221	17773
	Average	16550	16033	16176	17673
	Worst	16506	15995	16128	17583
	Std	20.2286	20.5269	22.0621	51.5802

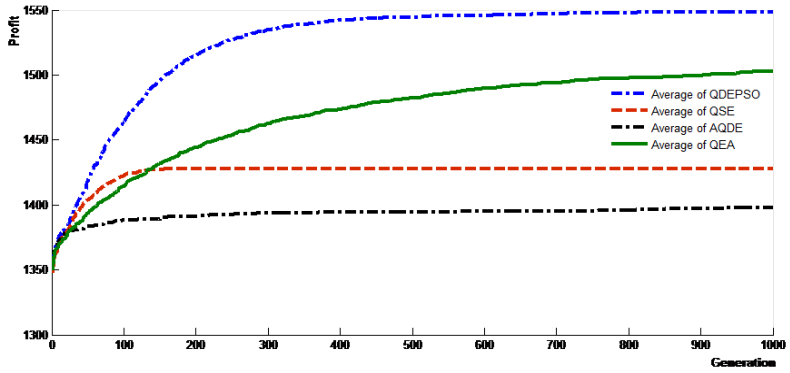


Figure 3.2: Average profits (250 items).

algorithm with the crossover operation (where the crossover control parameter $CR = 0.5$) and without the crossover operation ($CR = 0$). Moreover, we have com-

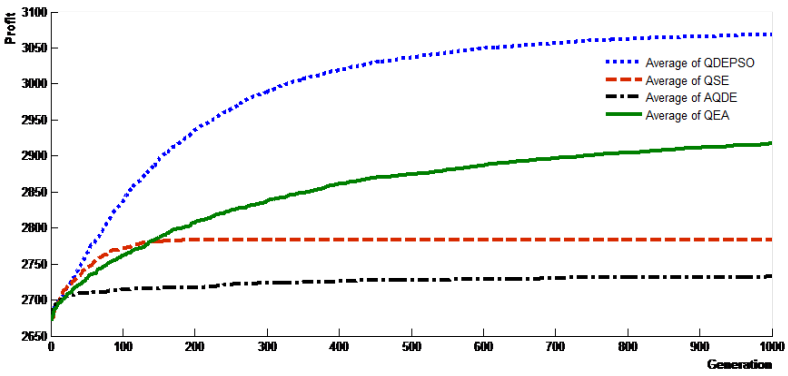


Figure 3.3: Average profits (500 items).

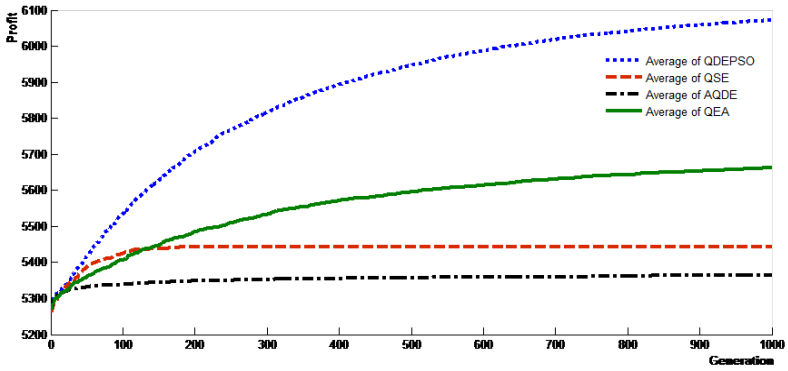


Figure 3.4: Average profits (1000 items).

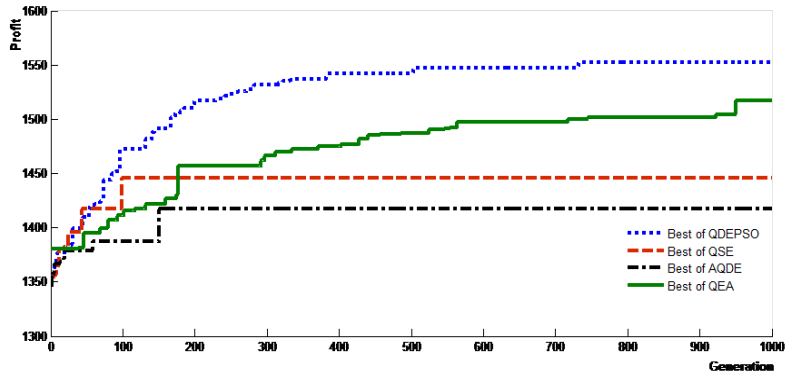


Figure 3.5: Best profits (250 items)

pared the DE crossover with other types of crossover, namely the one point and the two points crossover commonly used in genetic algorithms. The average, the best and the worst profits, the average time as well as the respective standard in

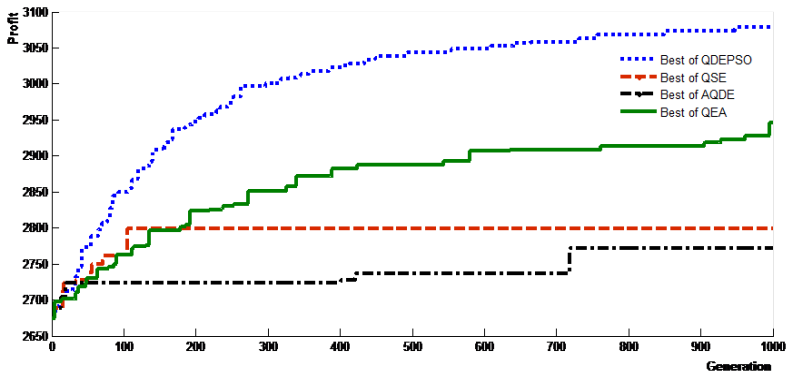


Figure 3.6: Best profits (500 items)

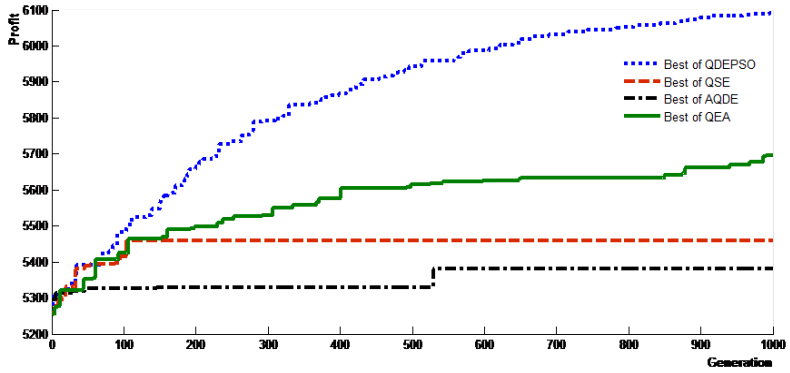


Figure 3.7: Best profits (1000 items)

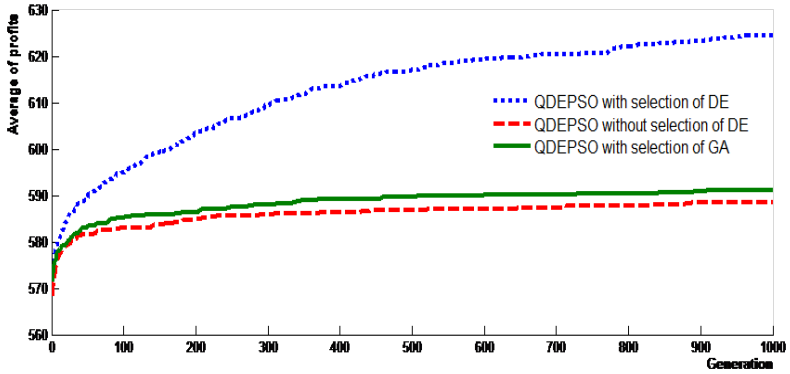


Figure 3.8: Average profits of QDEPSO algorithm with selection of DE, without selection of DE and with selection of GA (100 items).

deviations of 30 independent runs are presented in table 3.5. In experiment 2, we have used:

Table 3.4: The results of QDEPSO algorithm with selection of DE, without selection of DE, and with selection of GA on the 0-1 knapsack problem (experiment 1).

Item size		*QDEPSO with selection of DE*	*QDEPSO without selection of DE*	*QDEPSO with selection of GA*
100	Best	625	600	595
	Average	624.5	588.3333	590.9667
	Worst	620	580	585
250	Best	1544	1439	1444
	Average	1536	1422.3	1422
	Worst	1529	1409	1414
500	Best	3046	2716	2736
	Average	3038.8	2691.5	2696.2
	Worst	3031	2671	2671
1000	Best	6126	5426	5446
	Average	6111	5394.3	5405.7
	Worst	6096	5361	5381
1500	Best	9142	8072	8067
	Average	9109.3	8028	8044.8
	Worst	9087	7982	8012
2000	Best	12080	10725	10705
	Average	12035	10658	10669
	Worst	11990	10610	10645

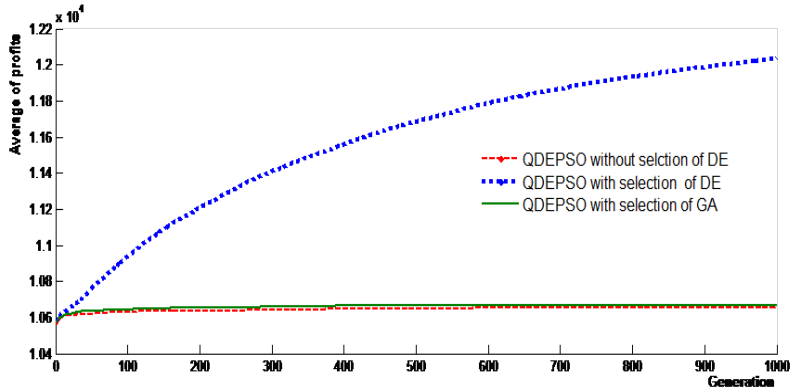


Figure 3.9: Average profits of QDEPSO algorithm with selection of DE, without selection of DE and with selection of GA (2000 items).

- The weights a_i, respective profits p_i and the knapsack capacity $C = \frac{3}{4}\sum_{i=1}^{m} a_i x_i$ defined by [Layeb 2013].
- Different size items of N varying from 50 to 3000.
- The maximum number of generations in all cases was chosen as 1000.

In order to prove the performance of the proposed algorithm in the global optimization compared to another class of the quantum-inspired algorithm, we have compared the QDEPSO algorithm with the quantum inspired harmony search algorithm (QIHSA) to solve the knapsack problem. The optimization results includ-

Table 3.5: The QDEPSO algorithm without crossover of DE, with DE crossover, with single point crossover of GA and with two point crossover of GA.

Items		*With DE crossover (Cr=0)*	*With DE crossover (Cr=0.5)*	*With Single-point crossover of GA*	*With two-point crossover of GA*
500	Average	2990.3	3039.5	3038.8	3038.3
	Best	3001	3046	3046	3046
	Worst	2981	3031	3031	3031
	Std	6.1752	3.5111	3.6397	3.1506
	Average time(s)	10.0748	11.4595	11.1594	11.5139
1000	Average	5893.8	6110.9	6102.5	6091.6
	Best	5926	6126	6121	6111
	Worst	5851	6096	6081	6076
	Std	18.0843	7.0091	9.0429	9.1142
	Average time(s)	19.9584	20.7725	21.3223	19.7728
1500	Average	8707.7	9112.8	9090.9	9063.2
	Best	8752	9137	9126	9112
	Worst	8647	9077	9042	9027
	Std	23.2206	15.7467	18.439	22.9071
	Average time(s)	29.8826	32.0788	30.6023	29.2978

ing the success rate (SR) and the best profit are presented in table 3.6 and figure 3.10.

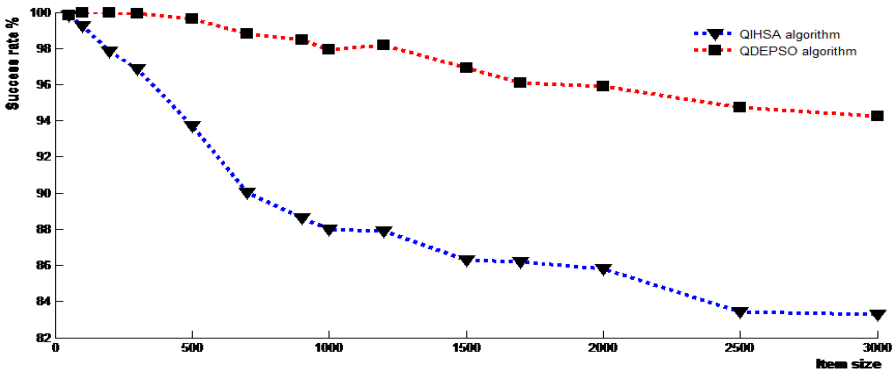


Figure 3.10: Success rate versus items' size.

The obtained results are discussed in the following points:

- The algorithm QDEPSO is more efficient for the high-dimensional 0–1 knapsack problems compared with QEA [Han 2002] , QSE [Wang 2007c], AQDE [Hota 2010] and QIHSA [Layeb 2013], the QDEPSO is the most efficient in the case where the number of items is too large. We illustrate, as an example, the cases where the size of items equals 2500 or 3000.

- Rapid exploration of the search space solutions: The QDEPSO converges

Table 3.6: The Results of QDEPSO and QIHSA on the 0-1 knapsack (experiment 2).

Test	Size	Optimal solution		QIHSA	QDEPSO
Knapinst50	50	1177	SR%	99.83	99.83
			best	1175	1175
Knapinst100	100	2466	SR%	99.27	100
			best	2448	2466
Knapinst200	200	4860	SR%	97.83	99.97
			best	4755	4859
Knapinst300	300	7137	SR%	96.88	99.91
			best	6915	7131
Knapinst500	500	11922	SR%	93.72	99.62
			best	11174	11877
Knapinst700	700	17107	SR%	90.02	98.79
			Best	15400	16901
Knapinst900	900	21269	SR%	88.62	98.48
			best	18850	20947
Knapinst1000	1000	24356	SR%	87.97	97.93
			best	21427	23854
Knapinst1200	1200	28617	SR%	87.91	98.2
			best	25160	28102
Knapinst1500	1500	35891	SR%	86.31	96.92
			best	30978	34787
Knapinst1700	1700	40609	SR%	86.18	96.09
			best	34998	39025
Knapinst2000	2000	49007	SR%	85.8	95.88
			best	42052	46990
Knapinst2500	2500	60613	SR%	83.43	94.72
			best	50 570	57418
Knapinst3000	3000	72371	SR%	83.28	94.27
			best	60273	68229

rapidly to global optimal. For example, the algorithm converges after the 400 iterations in the case of 250 items.

- High exploitation of the search space solutions: The algorithm continues the search around the global optimum i.e. the exploitation. For instance in the case of the 250 items, the QDEPSO converges to the optimal solution after iterating 400, but continues the research till the 600 *iterations*.

- According to the results of the average profits presented in figures 3.8-3.9, the tendency of the global convergence of QDEPSO algorithm including the DE selection is clearly increased. But, QDEPSO algorithm without the DE selection or with the roulette wheel selection of GA maintains a nearly constant profit due to its premature convergence.

- According to the comparison results presented in table 3.5, we conclude that the crossover operation plays an important role in exploring efficiently the search space. However, it is worth noticing that despite this importance, the

QDEPSO algorithm is not very sensitive to the kind of the used crossover. Indeed, in all cases, DE crossover gives slightly better results than the two other crossover operations in terms of best solution. This is interpreted by the fact that, in exploring the search space, the role of PSO remains the most decisive one even if it needs crossover to improve its outcome. Finally, the obtained results show the efficiency of the collaboration between the operations of mutation, DE crossover as well as the PSO update formulae in exploring the search space.

3.5 Chapter summary

In this chapter, we have proposed a new algorithm called QDEPSO to solve the global optimization problems. The proposed algorithm is a hybrid of three metaheuristic methods that are Differential Evolution Algorithm, Particle Swarm Optimization and Quantum-Inspired Evolutionary Algorithm. The QDEPSO uses the concepts of quantum computing such as the superposition state of the qubit and the quantum gate. Moreover, it uses the differentiation operations of the differential evolution algorithm such as mutation, crossover and selection. To better balance between the exploration and exploitation of the space of solutions' search, the algorithm adapts update formulas of the PSO in updating the population of solutions.

In order to validate the performance of our algorithm, we tested it for resolving the knapsack problem. The obtained results are better comparing with QEA [Han 2002] , AQDE [Hota 2010], QSE [Wang 2007c] and QIHSA [Layeb 2013] especially in the cases high dimensional instances of the 0-1 knapsack problem. The quality of the obtained results encourages us to implement the QDEPSO for other more complicated global optimization problems in the future.

CHAPTER 4

QIFAPSO for discrete optimization problems

Contents

4.1 Introduction

This chapter presents a new quantum-inspired firefly algorithm integrating the particle swarm movement to solve discrete combinatorial optimization problems. Potential solutions are represented in the quantum paradigm. We use then the key notions of this paradigm including superposition of Q-bit states, quantum measure and quantum gate. Moreover, the main idea that allows us to obtain a discrete variant of the firefly algorithm is that the movement of fireflies will no longer carried out directly on the "concrete fireflies", but rather by performing an

appropriate change in the rotational angle of the quantum solutions. This means that it is the probability to move in a given direction which will be subject of the movement changes.

Moreover, in order to determine the rotation angle, the algorithm may use, according to the context, either the movement formula of the original firefly algorithm ([Yang 2010a], [Yang 2009b]) or that used in the PSO algorithm [Kennedy 1995]. Finally, the proposed algorithm uses a variant of the Hamming distance to compute the distance between two discrete solutions. This measure is then used to update the attractiveness of fireflies.

The rest of the chapter is organized as follows. A brief recall about the multidimensional knapsack problem in section 4.2. Section 4.3 is devoted to the description of our algorithm QIFAPSO. All the relevant issues related to our algorithm are discussed, namely: the binary as well as the quantum representation of fireflies, the initialization of the population, the quantum measure, the method of computing the distance between discrete fireflies, the two movement strategies used (firefly and PSO strategies) and finally the complete pseudo-code of our algorithm. In section 4.4, we present and discuss a synthesis of various experimental tests done on the 0-1 knapsack problem both in its one dimensional and its multidimensional variants.

4.2 0–1 multidimensional knapsack problem: overview and related work

The multidimensional knapsack problem (MKP) is a constrained combinatorial optimization problem belonging to the class of NP-Complete problems [Kellerer 2004]. It is used to model several practical problems in which the aim is to maximize a profit whithin a limited number of resources, like the capital budgeting problem [Pendharkar 2006], project selection [Tavana 2013] and cryptography [Desmedt 2011]. Formally, the MKP is modelled as follows:

$$Maximize \sum_{j=1}^{n} p_j x_j \tag{4.1}$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq c_i,\ x_j \in \{0,1\}\ where\ 1 \leq j \leq n,\ 1 \leq i \leq m \tag{4.2}$$

with $p_j \geq 0$ *for* $1 \leq j \leq n$ *and* $0 \leq a_{ij} < b_i,\ b_i \geq 0$ *for* $1 \leq i \leq m,\ 1 \leq j \leq n$

$N = \{1,\ldots,n\}$ is the set of items, $M = \{1,\ldots,m\}$ is the set of available resources

with limited quantities, p_j is the profit brought by item j, a_{ij} is the consummation quantity of resource i by item j, b_i is the total quantity available of resource i. The objective of MKP is to select a subset of items that maximize the total profit while respecting the availability of each resource.

The multidimensional knapsack problem is the exemplary problem in constrained combinatorial optimization. Several resolution methods of the MKP have been proposed. Meta-heuristics are among the most successful approximated methods in solving the MKP since they found the best solutions in reasonable time. We can cite the genetic algorithm developed by [Chu 1998], the tabu search algorithm [Hanafi 1998], the ant colony optimization [Ke 2010] and the particle swarm optimization with repairing of infeasible solutions (PSO-R) [Liang 2010]. Recently, different discrete hybrid PSO methods have been proposed to solve the MKP including a modified binary particle swarm optimization (MBPSO) [Bansal 2012], particle swarm optimization with time-varying acceleration coefficients (BPSOTVAC) [Chih 2014] and binary accelerated particle swarm algorithm (BAPSA) [Beheshti 2013].

4.3 The proposed algorithm

In this section, we present our algorithm called Quantum-Inspired Firefly algorithm with Particle Swarm Optimization (QIFAPSO). We will use it to solve the multidimensional knapsack problem. The global architecture of the QIFAPSO algorithm is constituted of four main modules. The first module concerns the generation of the initial population of quantum fireflies (see section 4.3.3). The binary fireflies' solutions are constructed from the quantum ones by means of the quantum measure (see section 4.3.4) and they correspond to feasible solutions of the MKP.

After the binary fireflies are obtained from the quantum ones by quantum measures, the algorithm considers in turn each binary firefly and compares its brightness with that of all the other binary fireflies. The algorithm uses the second module in case the current binary firefly solution is less bright than the other one. This module corresponds to the quantum movement of a firefly towards a brighter firefly (see section 4.3.6). In the opposite case (the current binary firefly solution is brighter than the other one), the algorithm uses the third module. This module corresponds to the quantum movement of the firefly according to the movement strategy of particles used in the PSO approach, i.e., by taking into account the best position of the firefly as well as the best positions of the fireflies of its neighborhood (see section 4.3.7). Notice that both kinds of movements (FA and PSO) do

not operate directly on the current binary firefly but on the corresponding quantum firefly (the quantum firefly that has been used to construct the current binary firefly by a quantum measure). Then, the new quantum firefly resulting from the movement is used in the fourth module in order to construct a new binary firefly by quantum measure. This module contains also the evaluation of the objective function and the brightness update of the new binary firefly. Figure 4.1 depicts the global architecture of the proposed algorithm.

It is worth mentioning that our algorithm uses a star neighbor relation to coordinate between the different fireflies. Indeed each firefly is compared to all the other fireflies in terms of brightness. The result of this comparison determines whether a FA or a PSO movement strategy is applied. Moreover, when the PSO movement strategy is chosen, we consider again (at a lower level) a star neighbor relation since to determine the new position of the current firefly, we take into account both the position of the better solution found so far for this firefly and the position of the best solution known for the whole set of fireflies.

In summary, our algorithm is build on an interaction between the quantum and the binary representation levels of fireflies: the quantum fireflies are used initially to produce binary fireflies. Then the binary fireflies are compared between each others in terms of brightness. The result of the comparison between each couple of binary fireflies leads either a firefly or a PSO movement to be performed [1] . However, this movement is done first on the quantum level before using the new quantum firefly to produce a new binary firefly. After a movement, what is first updated is not the binary firefly but the probability to select each of its items in the next observation and consequently, the probabilities to select each of the possible binary fireflies superposed in the quantum firefly. Thus, using a quantum level allows one to preserve the population diversity. In addition, completing the FA movement by another kind of movement issued from the PSO approach, makes the new algorithm able to take advantage from both of these strategies and hence have a better exploration / exploitation of the search space.

4.3.1 Binary representation of fireflies

The choice of a representation for individuals is a crucial issue in any metaheuristic. The QIFAPSO algorithm uses the binary coding which is the most suitable for the selection of items. Each binary firefly solution Bf_i (the i^{th} firefly of

[1] In addition to the use of quantum representation, this constitutes a main point on which our algorithm differs from a classical firefly algorithm. Whilst the latter does not do any movement in case the current firefly is brighter than the other one, our algorithm suggests to use in this case a movement similar to that used in the PSO approach.

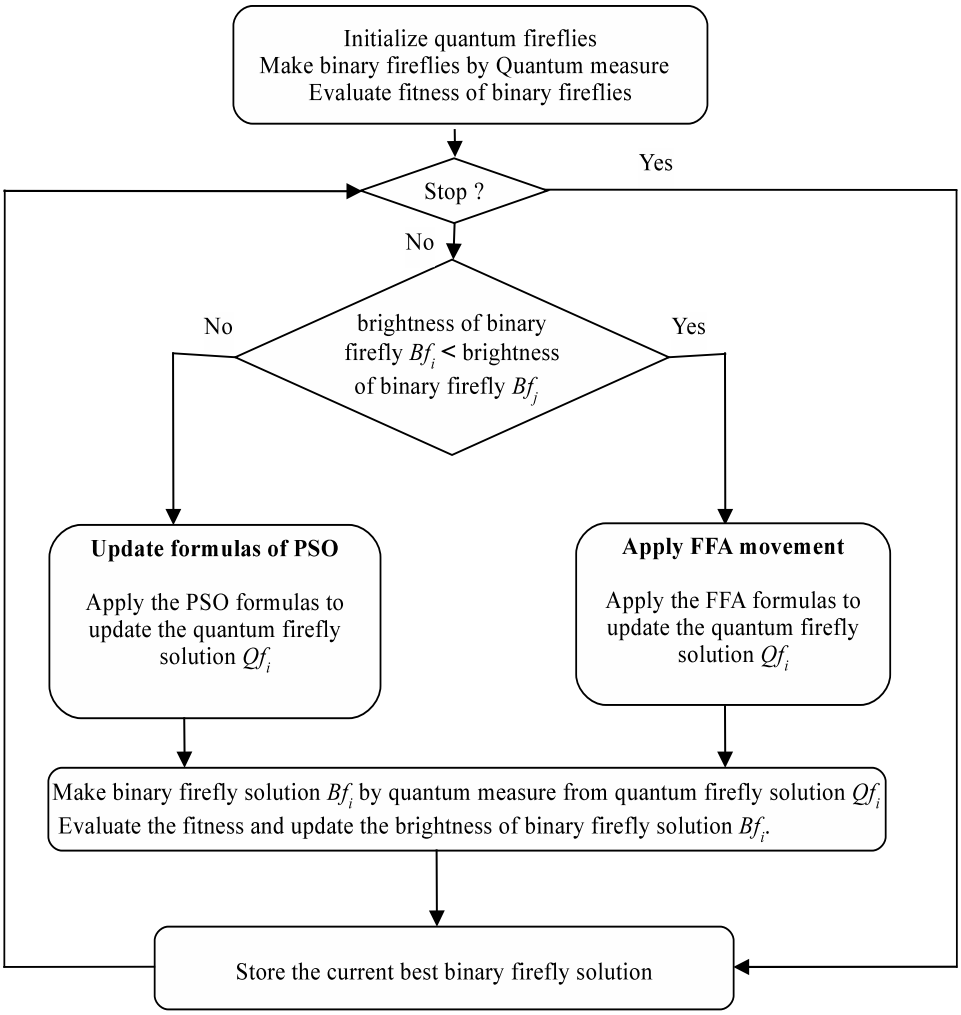


Figure 4.1: Global architecture of QIFAPSO algorithm

the binary population) is represented as a vector of length m (m is the size of the solution): $Bf_i = (Bf_{i1}, \ldots, Bf_{im})$.

$$\begin{cases} Bf_{ij} = 1 \; if \; the \; item \; x_j \; of \; Bf_i \; is \; selected \\ Bf_{ij} = 0 \; if \; the \; item \; x_j \; of \; Bf_i \; is \; rejected \end{cases}$$

For example, the binary representation of the selection of the first and the fourth item from the items set: $X = \{x_1, x_2, x_3, x_4\}$ is the vector: (1001).

$BF(t) = \{Bf_1^t, \ldots, Bf_n^t\}$ denotes the population of n binary fireflies' solutions at the t^{th} generation, where n is the size of the population; m is the length of each binary firefly solution Bf_i^t for $1 \leq i \leq n$.

4.3.2 Quantum representation of fireflies

We use the rotation angle in the Q-bit representation, where each quantum firefly solution Qf_i (the i^{th} quantum firefly in a quantum population) corresponds to a vector $\Theta_i = (\theta_{i1}, \ldots, \theta_{im})$ of variables θ_{ij} with $\theta_{ij} \in [0, \frac{\pi}{2}]$ for $(1 \leq j \leq m)$.

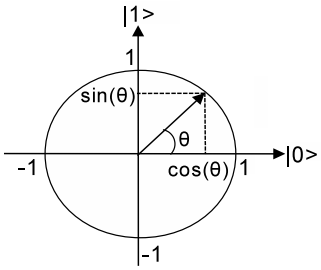


Figure 4.2: Q-bit representation by the unity circle

Each quantum firefly solution Qf_i is a string of quantum bits, calculated as follows:

$$Qf_i = \begin{bmatrix} cos(\theta_{i1}) & |cos(\theta_{i2}) & | & \ldots & | & cos(\theta_{im}) \\ sin(\theta_{i1}) & |sin(\theta_{i2}) & | & \ldots & | & sin(\theta_{im}) \end{bmatrix}$$

The probability amplitude of one quantum bit is defined with a pair of numbers $(cos(\theta_{ij}), sin(\theta_{ij}))$ where $|cos(\theta_{ij})|^2$ represents the probability of rejecting item x_j and $|sin(\theta_{ij})|^2$ represents the probability of selecting item x_j (in the i^{th} binary firefly). Of course $cos(\theta_{ij})$ and $sin(\theta_{ij})$ satisfy :
$|cos(\theta_{ij})|^2 + |sin(\theta_{ij})|^2 = 1$.

4.3.3 Initialization of quantum fireflies' population

In the initialization step, a quantum fireflies' population represents the linear superposition of all possible states with equal probability. In the case of knapsack problem, a quantum firefly represents a superposition of all possible combinations of selected items. For this, each vector Θ_i is initialized by:

$$\Theta_i^{t=0} = \left(\left(\frac{\pi}{4}\right), \ldots, \left(\frac{\pi}{4}\right)\right) \tag{4.3}$$

The Q-bit Qf_{ij} corresponding to θ_{ij} is probably located in the first quadrant of the unit circle. This initialization provides sufficiently high diversity in the initial population of quantum fireflies' solutions and gives an initial equilibrium between

the probability to select an item and the probability to reject it.

$$Qf_i^{t=0} = \begin{bmatrix} cos(\theta_{i1}^0) & |cos(\theta_{i2}^0) & | & \dots & | & cos(\theta_{im}^0) \\ sin(\theta_{i1}^0) & |sin(\theta_{i2}^0) & | & \dots & | & sin(\theta_{im}^0) \end{bmatrix} \tag{4.4}$$

for $1 \leq i \leq n$, where n is the number of quantum fireflies and m is the length of each quantum firefly solution Qf_i. Notice that we have, for $1 \leq i \leq n$ and $1 \leq j \leq m$, $|cos(\theta_{ij}^0)|^2 = \frac{1}{2}$ and $|sin(\theta_{ij}^0)|^2 = \frac{1}{2}$.

4.3.4 Quantum measure

In this operation, a quantum firefly solution is used to generate a binary firefly solution by Q-bits projection. For each Q-bit, represented by an angle Θ, we generate a random number r in the interval $[0,1]$. The corresponding bit in the binary firefly solution receives the value 1 if $|sin(\theta_{ij}^0)|^2 > r$, else the bit receives the value 0. Thus, a quantum solution encodes several binary solutions thanks to the superposition of qubit states. However, each qubit encodes a probability to select or to reject the corresponding item. At the quantum measure step, only one particular binary solution is extracted from the quantum one and this choice is guided by the probabilities encoded in qubits.

In the case of MKP, we distinguish two methods. The first one consists in observing all the Q-bits, then repairing the resulting sequence of bits to obtain a feasible solution. The second method allows one to create directly binary feasible solutions by quantum measure without repairing.

The construction of feasible binary fireflies' solutions by quantum measure is performed in two phases. In the first phase (ADD), the algorithm does not have initially any selected item. Then, it repeats several passes of measure operations until it finds an infeasible solution and in the last pass of the first phase, removes the item so that the solution stays feasible. In each pass, the algorithm browses randomly all the items and applies the quantum measure to each item which had not been yet selected in the precedent passes. In the second phase (ADD-2), the algorithm gives an additional chance to each item which respects the constraints but is not selected during the first phase to be observed again. Algorithm 4.1. below, constructs a feasible solution Bf by observing a quantum firefly Qf.

4.3.5 Distance between two binary fireflies

The QIFAPSO algorithm uses a variant of the Hamming distance. In general, the Hamming distance [Hamming 1950] allows one to quantify the difference between two sequences of symbols. In the case of the knapsack problem, the used distance

Algorithm 4.1 Observing a quantum firefly and constructing a feasible solution

Input: A_i: the accumulated resource of constraints.
Input: Qf: quantum firefly solution.
Output: Bf: the binary firefly solution.

Intialization:
$A_i \leftarrow 0, \forall i \in M = \{1,\ldots,m\}$;
Add phase 1:
while $(A_i \leq b_i)$ **do**
  #feasibility satisfied yet $\forall i \in M = \{1,\ldots,m\}$
  R $\leftarrow$ randpermut(n); #R is a vector resulting from the random permutation of the set $\{1,\ldots,n\}$ where n is the number of items
  $j \leftarrow 1$;
  while $((A_i \leq b_i) and (j \leq n))$ **do**
    if $((Bf[R[j]] = 0) and (rand() > cos(Qf[R[j]])^2))$ **then**
      $Bf[R[j]] \leftarrow 1$; #the selection of an item not selected in the precedent pass by the quantum measure
      $A_i \leftarrow A_i + a_{iR[j]}$;
    end if
    $j \leftarrow j + 1$;
  end while
end while
$Bf[R[j-1]] \leftarrow 0$; #remove the last $(j-1)^{th}$ item so that the solution stays feasible
$A_i \leftarrow A_i - a_{iR[j-1]}$;

Add phase 2:
for $(j := 1 : n)$ **do**
  if $((Bf[R[j]] = 0)\ and\ (A_i + a_{ij} \leq b_i)\ and\ (rand() > cos(Qf[R[j]])^2))$ **then**
    $Bf[j] \leftarrow 1$; #the selection of an item not selected in the precedent pass and respecting all the constraints
    $A_i \leftarrow A_i + a_{ij}$;
  end if
end for

represents the number of distinct bits between two binary solutions. The proposed distance gives the ratio between the Hamming distance and the number of items that are not selected in the two binary fireflies' solutions.

$$d(Bf_{ik}, Bf_{jk}) = \begin{cases} 1 & \text{if } Bf_{ik} \neq Bf_{jk} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

$$DHamming(Bf_i, Bf_j) = \sum_{k=1}^{m} d(Bf_{ik}, Bf_{jk}) \tag{4.6}$$

$$Distance(Bf_i, Bf_j) = \frac{DHamming(Bf_i, Bf_j)}{2m - (\sum_{k=1}^{m} Bf_{ik} + \sum_{k=1}^{m} Bf_{jk})} \tag{4.7}$$

where m is the size of a binary firefly solution.

Two binary fireflies' solutions Bf_i and Bf_j are said to be identical if and only if $Distance(Bf_i, Bf_j) = 0$ and $Bf_{ik} = Bf_{jk}$ for $1 \leq k \leq m$. Accordingly, they are said to be completely different if and only if $Distance(Bf_i, Bf_j) = 1$ and $Bf_{ik} \neq Bf_{jk}$ for $1 \leq k \leq m$.

The following example shows the utility of the distance proposed compared with the classical Hamming distance. Consider two binary fireflies Bf_i and Bf_j of length 6 (containing six bits) where four couples of bits are different and two couples of bits are identical (see figure 4.3). The classical Hamming distance does not distinguish between the two following configurations: The two identical couples equal 1 or equal 0, i.e., the attractiveness is considered identical in the both cases. However, the proposed distance distinguishes between these two cases. Indeed, the proposed distance considers that the attractiveness is greater in the case where the two identical couples equal 0 than in the other case.

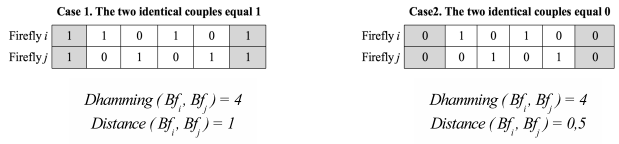


Figure 4.3: Discrete distance between two binary fireflies' solutions

4.3.6 Quantum movement according to the firefly algorithm strategy

Let Qf_i and Qf_j be two quantum fireflies' solutions such that Qf_i is the current quantum firefly solution and Qf_j is brighter than Qf_i. Then, Qf_i is attracted by Qf_j. Its movement is determined as follows:

$$\theta_{ik}^{t+1} = \begin{cases} \theta_{ik}^{t} + \beta_0 e^{-\gamma r_{ij}^2}(\theta_{jk}^{t} - \theta_{ik}^{t}) + \alpha_t \varepsilon_{ik}^{t} & \text{if } Bf_{ik}^{t} \neq Bf_{jk}^{t} \\ \theta_{ik}^{t} & \text{otherwise} \end{cases} \tag{4.8}$$

- θ_{ik}^{t}, θ_{ik}^{t+1} are the angles of the k^{th} Q-bit for the quantum firefly Qf_i at the iterations t and $t+1$ respectively.
- $\beta_0 e^{-\gamma r_{ij}2}$ represents the attractiveness between the two binary fireflies' solutions Bf_i^t and Bf_j^t at the iteration t.

- α_t is a random parameter which may be constant and ε^t is a vector of random numbers uniformly distributed in the interval $[0,1]$.

The effect of the quantum movement of a Q-bit of a firefly is to increase or decrease the selection probability of the corresponding item in the binary firefly. In order to explore the search space in an efficient way, this quantum movement is guided as follows : if the k^{th} bit of the binary firefly Bf_i is different from that of the firefly Bf_j then we replace the value of the corresponding Q-bit of firefly Qf_i by the new value resulting from the movement, else we conserve the state of the Q-bit of firefly Qf_i.

The new quantum firefly Qf_i^{t+1} is created by changing the rotation angle according to the FFA movement strategy (presented in equation 4.8). This change is given by:

$$Qf_i^{t+1} = \begin{bmatrix} cos(\theta_{i1}^{t+1}) & |cos(\theta_{i2}^{t+1}) & | & \dots & | & cos(\theta_{im}^{t+1}) \\ sin(\theta_{i1}^{t+1}) & |sin(\theta_{i2}^{t+1}) & | & \dots & | & sin(\theta_{im}^{t+1}) \end{bmatrix} \tag{4.9}$$

Let us take an example to illustrate the quantum movement according to the firefly algorithm strategy. Suppose that the current binary firefly considered in the algorithm is $Bf_i = (0,1,1,0,1)$ and that this firefly is compared to the binary firefly $Bf_j = (1,1,0,0,0)$ to decide what kind of movement has to be used at this stage. Suppose that Bf_i turns out to be less bright than B_j (recall that the brightness of a firefly depends on the value of its objective function). Then, the FA movement strategy is used. Notice that in the opposite case (if Bf_i was brighter than B_j), it is the PSO movement strategy that would have been used. Now, the variant of the Hamming distance as well as the attractiveness between Bf_i and Bf_j are calculated giving the values : $r_{ij} = 0.6$ and $\beta = 0.9646$ respectively. These values are then used to apply the movement formula given in equation 4.8. on the two quantum vectors $\Theta_i = (0.01\pi, 0.03\pi, 0.5\pi, 0.4\pi, 0.01\pi)$ and $\Theta_j = (0.1\pi, 0.15\pi, 0.1\pi, 0.01\pi, 0.3\pi)$ corresponding to quantum fireflies Qf_i and Qf_j from which the binary fireflies Bf_i and Bf_j have been observed respectively. The new obtained vector for Θ_i is $(0.075\pi, 0.17\pi, 0.14\pi, 0.02\pi, 0.31\pi)$. Thus, the quantum movement is guided by the values of the two binary fireflies but it affects the quantum fireflies. For each qubit of the quantum firefly, the effect of the movement consists in possibly increasing or decreasing the probability of selecting the corresponding item in next observations.

4.3.7 Quantum movement according to the PSO strategy

Let Qf_i and Qf_j be two quantum fireflies' solutions such that Qf_i is the current quantum firefly solution and Qf_i is brighter than Qf_j. In this case, the movement of the quantum firefly solution Qf_i is performed according to the PSO strategy. It is done by changing the rotation angle from θ_{ik}^t to θ_{ik}^{t+1} for each Q-bit. To determine the rotation angle of each Q-bit, we adapt the equations 2.19 and 2.20 for the update of velocity and position of a particle as presented by equations 4.10 and 4.11 below (note that m is the size of a solution and $1 \leq k \leq m$):

$$\Delta\theta_{ik}^{t+1} = \omega\Delta\theta_{ik}^t + c_1 r_1(\theta b_{ik}^t - \theta_{ik}^t) + c_2 r_2(\theta g_k^t - \theta_{ik}^t) \tag{4.10}$$

$$\theta_{ij}^{t+1} = \begin{cases} \theta_{ik}^t + \Delta\theta_{ik}^{t+1} & \text{if } Bf_{ik} \neq Fg_k^t \text{ or } Bf_{ik} \neq Fb_{ik}^t \\ \theta_{ij}^t & \text{otherwise} \end{cases} \tag{4.11}$$

- $\Delta\theta_{ik}^t$ and $\Delta\theta_{ik}^{t+1}$ are the rotation angles of the k^{th} Q-bit for the quantum firefly solution Qf_i at the iterations t and $t+1$ respectively, with $\Delta\theta_{ik}^t$ is initialized to 0 for $t = 0$.
- Bf_{ik}^t is the k^{th} bit of the binary firefly solution Bf_i^t at the iteration t.
- Fb_{ik}^t is the k^{th} bit of the best binary firefly solution Fb_i^t at the iteration t.
- Fg_k^t is the the k^{th} bit of the best binary firefly solution of the binary fireflies' population at the iteration t.
- θ_{ik}^t and θ_{ik}^{t+1} are the rotation angles of the k^{th} Q-bit of the quantum firefly solution Qf_i at the iterations t and $t+1$ respectively.
- θb_{ik}^t is the rotation angle of the k^{th} Q-bit of the best quantum firefly solution Qf_i.
- θg_k^t is the rotation angle of the k^{th} Q-bit of the best quantum firefly solution in the population at the iteration t.
- r_1 and r_2 are random numbers in the interval $[0, 1]$.
- w is the inertia weight, c_1 and c_2 are the learning factors.

The new quantum firefly Qf_i^{t+1} is created by changing the rotation angle according to the PSO movement strategy (presented in equations 4.10 and 4.11). This change is given by:

$$Qf_i^{t+1} = \begin{bmatrix} cos(\theta_{i1}^{t+1}) & |cos(\theta_{i2}^{t+1}) & | & \dots & | & cos(\theta_{im}^{t+1}) \\ sin(\theta_{i1}^{t+1}) & |sin(\theta_{i2}^{t+1}) & | & \dots & | & sin(\theta_{im}^{t+1}) \end{bmatrix} \tag{4.12}$$

4.3.8 QIFAPSO algorithm

Now, we are ready to give the pseudo-code of the algorithm QIFAPSO (see Algorithm 4.2.).

4.4 Experimental results

In this section, the performance of QIFAPSO algorithm is assessed with a large number of experiments. These experiments are carried out on a laptop equipped with an Intel Core (TM) I5-3230M processor and implemented in MATLAB. To check the efficiency of our algorithm in solving constrained combinatorial optimization problems, we applied QIFAPSO to solve the knapsack problem both in its simple and multi-constraint (multi-dimensional) versions.

4.4.1 0-1 Simple Knapsack Problem

In this experiment, we have used test instances of simple knapsack that are randomly generated. The weights a_i and the profits p_i are computed as follows [Han 2002]:

- a_i is a random integer value generated uniformly in the set $\{1,\dots,10\}$.
- The profit p_i is given by $p_i = a_i + 5$ and the knapsack capacity b is given by: $b = \frac{1}{2}\sum_{i=1}^{n} a_i x_i$.
- We have used various sizes (n) ranging from 1000 to 3000. The maximum number of generations is put to 1000.

QIFAPSO algorithm is compared with the quantum inspired algorithms: QEA [Han 2002], AQDE [Hota 2010] and QSE [Wang 2007c] in solving the knapsack problem. We have chosen the comprison with the class of quantum inspired algorithms in order to verify the relevance of integrating the concepts of quantum computation to the firefly algorithm.The parameters used for the preceding algorithms are presented in the following table where the population size is 10. The average, the best and the worst profits, as well as, the respective standard in deviations of 30 independent runs are presented in table 4.2 and in figures 4.4-4.7.

Algorithm 4.2 QIFAPSO Algorithm

Initialize the parameters of the algorithm: β_0, γ, c_1, c_2 and ω.
Initialize the quantum fireflies' population: $QF(0) = \{Qf_1^0, \ldots, Qf_n^0\}$ by equations 4.3 and 4.4.
Construct the binary fireflies' population: $BF(0) = \{Bf_1^0, \ldots, Bf_n^0\}$ from $QF(0)$ by algorithm 4.1.
Evaluate the fitness of the binary fireflies' population $BF(0)$ which is directly proportional to brightness.
Store the best binary fireflies' solutions among $BF(0)$ into $BFB(0)$ and the best quantum fireflies' solutions among $QF(0)$ into $QFB(0)$.
Store the current best binary firefly solution among $BFB(0)$ into Fg and the current best quantum firefly solution among $QFB(0)$ into QFg.
t := 1;
while $(t \leq Max\_generations)$ **do**
  for $(i = 1 : n$: all quantum fireflies$)$ **do**
    for $(j = 1 : n$: all quantum fireflies$)$ **do**
      if $(I(Bf_i^t) < I(Bf_j^t))$ **then**
        #I is the brightness of a binary firefly solution
        Compute the discrete distance between binary fireflies' solutions Bf_i^t and Bf_j^t (equations 4.5 and 4.6).
        Compute the attractiveness between binary fireflies' solutions Bf_i^t and Bf_j^t (equation 2.16).
        Compute the new position of the quantum firefly solution Qf_i^{t+1} by moving the quantum firefly Qf_i^t towards the quantum firefly Qf_j^t by applying the formulas 4.8 and 4.9.
      else
        #$(I(Bf_i^t) \geq I(Bf_j^t))$
        Construct the new position of the quantum firefly solution Qf_i^{t+1} by applying the PSO formulas (equations 4.10-4.11).
      end if
      Construct the new binary firefly solution Bf_i^{t+1} from the quantum firefly solution Qf_i^{t+1} by algorithm 4.1.
      Evaluate the fitness of the new binary firefly solution Bf^{t+1} which is directly proportional to its brightness.
      #Store the best binary firefly solution Bfb_i^{t+1} and the best quantum firefly solution Qfb_i^{t+1}
      if $I(Bfb_i^t) < I(Bf_i^{t+1})$ **then**
        $Bfb_i^{t+1} \leftarrow Bf_i^{t+1}$
        $Qfb_i^{t+1} \leftarrow Qf_i^{t+1}$
      end if
      #Store the current best binary firefly solution Bfg_i^{t+1} and the current best quantum firefly solution Qfg_i^{t+1}
      if $I(Bfg_i^t) < I(Bf_i^{t+1})$ **then**
        $Bfg_i^{t+1} \leftarrow Bf_i^{t+1}$
        $Qfg_i^{t+1} \leftarrow Qf_i^{t+1}$
      end if
    end for
  end for
  $t \leftarrow t + 1$
end while

Table 4.1: Parameters of the algorithms

Algorithm	Parameters
QEA	Quantum rotation angle $\Delta\theta = 0.01\pi$
QSE	Quantum rotation angle $\Delta\theta = 0.01\pi$, $w = 0.7298$, $c_1 = 1.42$, $c_2 = 1.57$
AQDE	Differentiation Control (CF) = random(0,1) $\times$ random(0, 1) $\times$ 0.1, Crossover control (CR) = rand('norm', 0.5, 0.0375): generates a random number from the Gaussian distribution, $\mu = 0.5$, $\sigma = 0.0375$
QIFAPSO	$\alpha = 0.2$, $\beta_0 = 1$, $\gamma = 0.1$, $w = 0.9$, $c_1 = c_2 = 1$

Table 4.2: Comparison of QIFAPSO with QEA, QSE and AQDE on the 0-1 knapsack problem.

item size		QEA	QSE	AQDE	QIFAPSO
1000	Best	5695	5460	5382	6151
	Ave	5662.5	5442.2	5364.4	6134.5
	Worst	5633	5422	5342	6116
	Std	12.7028	11.2975	10.1018	7.6733
1500	Best	8464	8128	8198	9217
	Ave	8439.4	8082.8	8178.7	9198.4
	Worst	8414	8039	8149	9177
	Std	15.1535	20.6722	13.6188	9.2083
2000	Best	11217	10813	10951	12305
	Ave	11191	10781	10900	12276
	Worst	11164	10747	10865	12245
	Std	14.7202	16.2827	24.6167	14.0367
2500	Best	13907	13466	13569	15364
	Ave	13866	13394	13523	15314
	Worst	13839	13342	13482	15278
	Std	19.0504	24.5971	23.5438	18.9084
3000	Best	16604	16071	16221	18384
	Ave	16550	16033	16176	18334
	Worst	16506	15995	16128	18284
	Std	20.2286	20.5269	22.0621	27.1565

From table 4.2 and figures 4.4-4.7, we can point out that the results obtained by QIFAPSO algorithm are clearly better than those obtained by the quantum-inspired algorithms: QEA, QSE and AQDE, particularly in the case where high dimensional instances of the 0-1 knapsack are used. This shows that comparing with these algorithms, QIFAPSO ensures a better equilibrium between exploration and exploitation of the search space.

4.4.2 Multidimensional Knapsack Problem

In this section the performances of QIFAPSO algorithm are evaluated on the multi-dimensional knapsack problem in order to show its efficiency in dealing with constrained combinatorial optimization. We have conducted for that purpose several

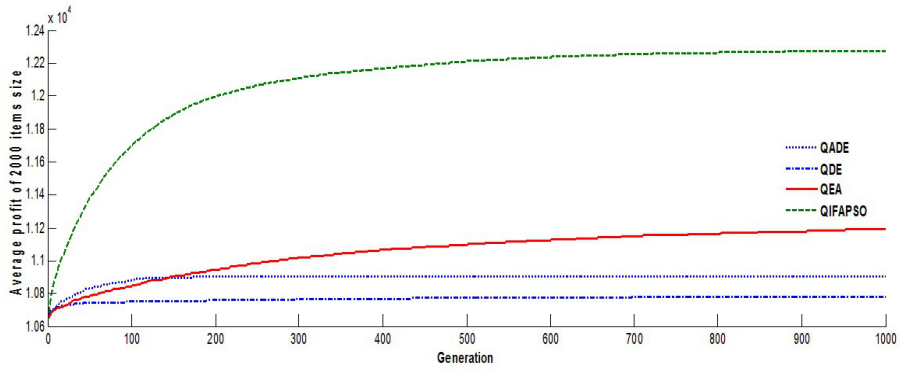


Figure 4.4: Average profits (2000 items)

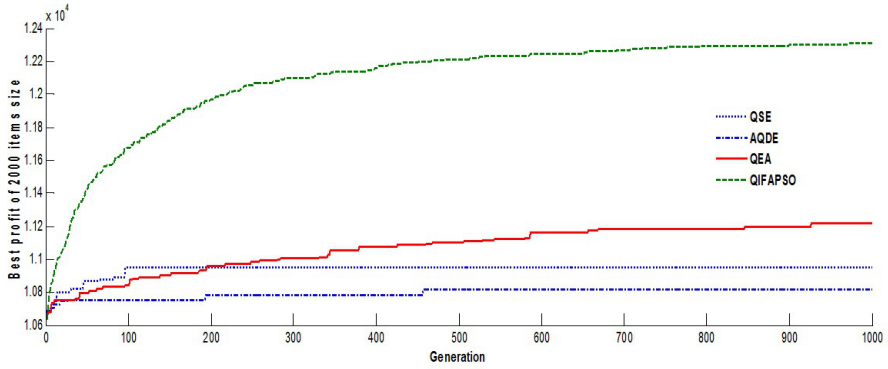


Figure 4.5: Best profits (2000 items)

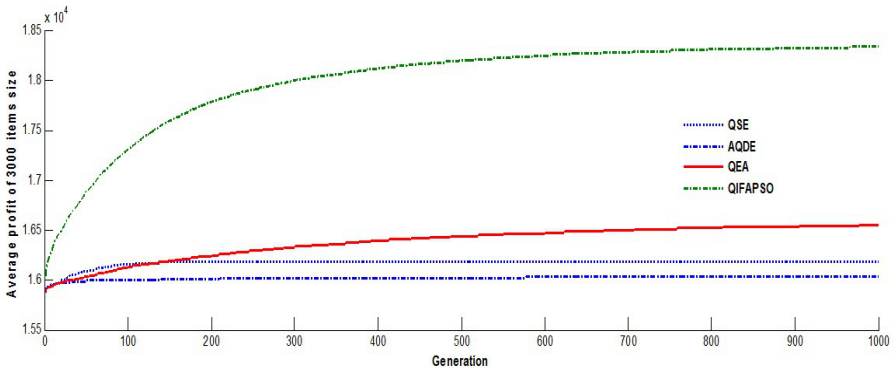


Figure 4.6: Average profits (3000 items)

experiments using instances of the benchmark library: OR-library [Beasley 2005].

In the first experiment, we have performed an (internal) comparison between three versions of QIFAPSO algorithm: QIFAPSO without the repairing opera-

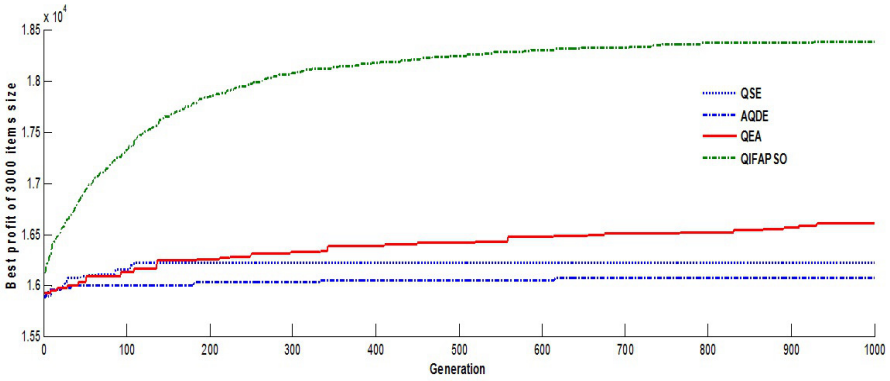


Figure 4.7: Best profits (3000 items)

tion, QIFAPSO without the PSO movement and QIFAPSO with repairing and PSO movement.

For the repair operator, we have used that presented in [Horng 2012]. This operator is based on the profit density ratio of every item given by : $u_i = \frac{p_i}{\sum_{j=1}^{m} a_{ij}}$. This operator is performed in two phases: in the first phase (DROP phase), the items with the lowest profit density are removed in turn until a feasible solution is found. Then, in the second phase (ADD phase), items with the highest profit density are added until one constraint is violated.

We have done five running tests both for the *maknapcb1 benchmark* (100 items and 5 constraints) and for *maknapcb4 benchmarks* (100 items and 10 constraints). To assess the performance of the three versions of the proposed algorithm, we have used the average error criteria defined in [Beheshti 2013]:

$$AE = \frac{1}{N} \sum_{i=1}^{N} \frac{t_i - y_i}{t_i} \times 100 \tag{4.13}$$

where N is the number of used benchmarks, t_i is the maximum profit known for benchmark i and y_i presents the best, the average or the worst result obtained for benchmark i by the three versions of QIFAPSO algorithm. The best, the worst and the average profit as well as the average error are depicted in table 4.3 and figures 4.8 and 4.9.

According to table 4.3 and figures 4.8 and 4.9, we can remark that QIFAPSO algorithm with the repairing operation and the PSO movement outperforms QIFAPSO without repairing or without PSO. Indeed, with (complete) QIFAPSO, the average error of the best profit we obtain is minimal (AE = 0.09%). Moreover, QIFAPSO without repairing gives an important improvement between the initial solution and the final one. Finally, we conclude that injecting the PSO movement

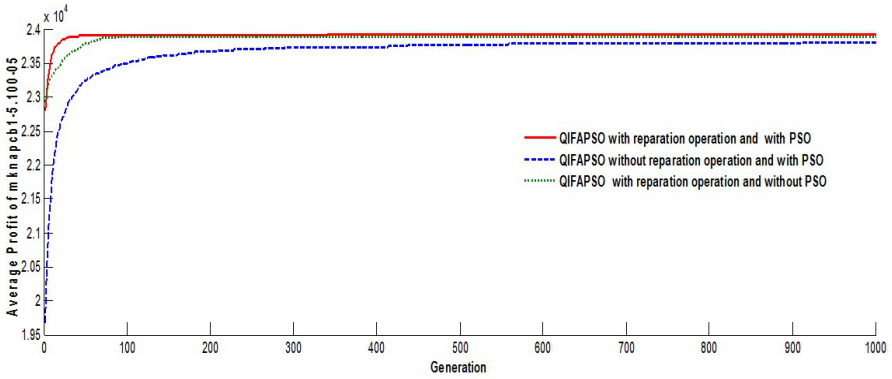


Figure 4.8: Performances of QIFAPSO algorithm without repairing, without PSO and with repairing and PSO on mknapcb 1-5.100-05

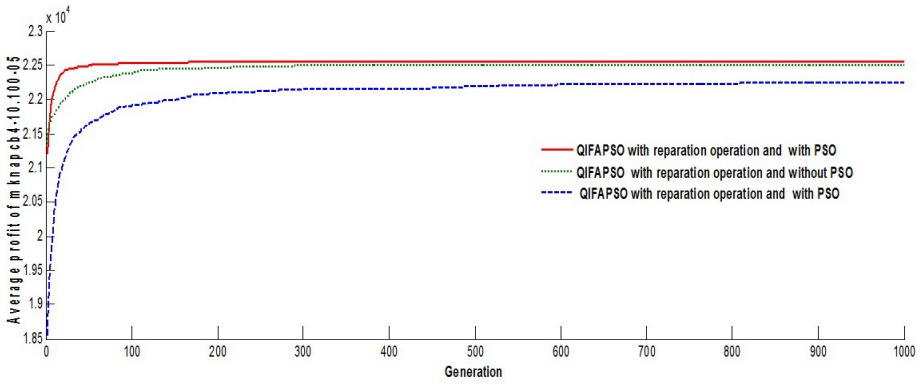


Figure 4.9: Performances of QIFAPSO algorithm without repairing, without PSO and with repairing and PSO on mknapcb 4-10.100-05

in the Firefly algorithm improves the performance of the algorithm in exploring the search space.

Another experiment is performed to compare QIFAPSO algorithm with PSO-based algorithms, namely: MBPSO [Bansal 2012], BPSOTVAC [Chih 2014] and CBPSOTVAC [Chih 2014] for solving the multidimensional knapsack problem. We have used for that the five evaluation criteria proposed in [Chih 2014]: The success ratio (SR), the mean absolute deviation (MAD), the mean absolute percentage of error (MAPE), the minimal error (LE) and the standard deviation of the solution (SD). As stated in [Chih 2014], SR refers to the number of runs that produce the optimum solution within the termination criterion. MAD is the average of the absolute difference between the simulation data and the given optimal solution. MAPE is obtained by dividing MAD by the corresponding optimal solution. It expresses the accuracy as a percentage. LE is the least error obtained by the mini-

Table 4.3: Performances of QIFAPSO algorithm without repairing, without PSO and with repairing and PSO (mknapcb1, mknapcb4).

Benchmark	Best	Max profit	QIFAPSO	QIFAPSO without repairing operation	QIFAPSO without PSO movement
mknapcb1-5.100-01	24381	Best	**24381**	24329	**24381**
		Ave	**24312**	24116	24248
		Worst	**24104**	23422	24062
mknapcb1-5.100-02	24274	Best	**24274**	24225	**24274**
		Ave	**24206**	23946	24201
		Worst	**24116**	23308	24054
mknapcb1-5.100-03	23551	Best	**23538**	23551	23523
		Ave	**23513**	23378	23497
		Worst	**23444**	23153	23422
mknapcb1-5.100-04	23534	Best	**23484**	23390	23484
		Ave	**23451**	23225	23370
		Worst	**23354**	23016	23201
mknapcb1-5.100-05	23991	Best	**23991**	23947	23966
		Ave	**23924**	23805	23888
		Worst	**23793**	23485	23782
mknapcb4-10.100-01	23064	Best	**23057**	23044	23025
		Ave	**22954**	22749	22839
		Worst	**22788**	22262	22689
mknapcb4-10.100-02	22801	Best	**22750**	22678	22743
		Ave	**22649**	22368	22589
		Worst	**22508**	21707	22477
mknapcb4-10.100-03	22131	Best	**22131**	21942	22044
		Ave	**21977**	21717	21895
		Worst	**21783**	21185	21707
mknapcb4-10.100-04	22772	Best	**22772**	22635	22631
		Ave	**22566**	22373	22487
		Worst	**22339**	21990	22261
mknapcb4-10.100-05	22751	Best	**22654**	22499	22693
		Ave	**22555**	22249	22495
		Worst	**22320**	21786	22321
Avg. error of best profit (%)			**0.09 %**	0.44 %	0.21 %
Avg. error of Ave profit (%)			**0.50 %**	1.44 %	0.76 %
Avg. error of worst profit (%)			**1.17 %**	3.41 %	1.41 %

mum of absolute difference between the optimum solution and final solution. SD is the standard deviation of final solutions over runs.

We have compared QIFAPSO algorithm with the three algorithms: MBPSO, BPSOTVAC and CBPSOTVAC on two classes of MKP benchmarks: A first class of low-dimensional knapsack problems containing instances of SENTO, WEING and WEISH and a second class of high dimensional knapsack problems containing instances whose size is ranging from 250 to 500 items and the number of constraints is between 5 and 10. The results of the comparison are presented in tables 4.4-4.6 and figures 4.10-4.11. The results show that our algorithm is more efficient than the binary PSO algorithms for the first class of low-dimensional knapsack instances. Indeed, our algorithm outperforms the other algorithms on the different

Table 4.4: Comparison between QIFAPSO algorithm and MBPSO, BPSOTVAC and CBPSOTVAC algorithms on low-dimensional knapsack instances (Sento, Weing).

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Sento1	30	60	MBPSO	0.16	44.81	0.0058	229	43.23
			BPSOTVAC	0.57	8.74	0.0011	34	11.52
			CBPSOTVAC	0.39	136.28	0.021	3146	357.78
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Sento2	30	60	MBPSO	0.03	24.85	0.0029	81	18.8
			BPSOTVAC	0.27	9.42	0.001	38	7.04
			CBPSOTVAC	0.2	53.53	0.0063	633	101.03
			QIFAPSO	**0.44**	**1.8**	**0.0002**	**1**	**3.2814**
Weing1	2	28	MBPSO	0.82	110.79	0.0008	801	250.43
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.92	51.25	0.0004	1961	281.98
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weing2	2	28	MBPSO	0.65	117.45	0.0009	1700	314.08
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.88	123.19	0.0009	3341	545.5
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weing3	2	28	MBPSO	0.11	1053.2	0.0112	3500	876.78
			BPSOTVAC	0.92	6.42	0.00007	160	25.53
			CBPSOTVAC	0.75	173.07	0.0019	3789	672.42
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weing4	2	28	MBPSO	0.76	570.6	0.0049	4001	1270.8
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.97	42.83	0.0004	3774	378.58
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weing5	2	28	MBPSO	0.52	1629.21	0.017	4778	1923.5
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.94	85.62	0.0009	4728	572.82
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weing6	2	28	MBPSO	0.36	310.2	0.0023	1340	322.4
			BPSOTVAC	0.97	11.7	0.00009	390	66.86
			CBPSOTVAC	0.87	91.71	0.0007	2460	343.45
			QIFAPSO	**0.97**	**11.7**	**0.00009**	**390**	**66.86**
Weing7	2	105	MBPSO	0.02	660.86	0.0006	6111	1130.6
			BPSOTVAC	0	281.23	0.00026	2069	383.74
			CBPSOTVAC	0	11272.9	0.011	154486	30020
			QIFAPSO	**0.45**	**70.67**	**0.000064**	**63**	**285.6132**
Weing8	2	105	MBPSO	0.03	5824.7	0.0095	44731	4704.3
			BPSOTVAC	0.35	1872.44	0.003	6463	2000.9
			CBPSOTVAC	0.2	27128.4	13.7	623862	75169
			QIFAPSO	**0.98**	**51.76**	**8.290**	**1943**	**375.5081**

evaluation criteria with a significant difference. For the class of high-dimensional knapsack instances, according to anova statistical test, we remark clearly that our algorithm is more efficient than the other algorithms BPSOTVAC and CBPSOTVAC.

Table 4.7 gives the best results obtained for mknapcb1-5.100 and mknapcb1-5.500 knapsack benchmarks by QIFAPSO and the PSO methods : self-adapted check repair operator-based PSO (SACRO-BPSO-TVAC, SACRO-CBPSO-TVAC)

Table 4.5: Comparison between QIFAPSO algorithm and MBPSO, BPSOTVAC and CBPSOTVAC algorithms on low-dimensional knapsack instances (Weish).

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish1	5	30	MBPSO	0.82	10.9	0.0024	114	26.340
			BPSOTVAC	1	0	0	0	0.000
			CBPSOTVAC	0.94	5.45	0.0012	248	32.810
			QIFAPSO	**1**	**0**	**0**	**0**	**0.000**
Weish2	5	30	MBPSO	0.55	8.39	0.0018	123	18.010
			BPSOTVAC	0.64	1.8	0.0004	5	2.410
			CBPSOTVAC	0.66	4.12	0.0009	231	23.120
			QIFAPSO	**0.96**	**0.2**	**0.000044**	**5**	**0.9847**
Weish3	5	30	MBPSO	0.63	20.54	0.0051	141	34.980
			BPSOTVAC	0.99	0.63	0.00015	63	6.300
			CBPSOTVAC	0.95	9.21	0.0024	394	52.690
			QIFAPSO	**1**	**0**	**0**	**0**	**0.000**
Weish4	5	30	MBPSO	0.96	1.76	0.0004	56	8.990
			BPSOTVAC	1	0	0	0.000	
			CBPSOTVAC	0.99	8.59	0.0023	859	85.900
			QIFAPSO	**1**	**0**	**0**	**0**	**0.000**
Weish5	5	30	MBPSO	0.99	0.54	0.00012	54	5.400
			BPSOTVAC	1	0	0	0.000	
			CBPSOTVAC	0.98	8.11	0.0021	742	74.450
			QIFAPSO	**1**	**0**	**0**	**0**	**0.000**
Weish6	5	40	MBPSO	0.32	15.36	0.0028	56	14.390
			BPSOTVAC	0.59	6.68	0.00121	18	8.190
			CBPSOTVAC	0.53	23.21	0.0044	518	79.280
			QIFAPSO	**0.8**	**2.6**	**0.00046**	**13**	**5.226**
Weish7	5	40	MBPSO	0.64	10.2	0.0018	122	18.920
			BPSOTVAC	0.96	0.7	0.00013	18	3.450
			CBPSOTVAC	0.78	19.17	0.0036	511	71.950
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weish8	5	40	MBPSO	0.44	7.24	0.0013	72	13.070
			BPSOTVAC	0.79	0.42	0.00008	2	0.820
			CBPSOTVAC	0.68	8.84	0.0016	418	42.810
			QIFAPSO	**1**	**0**	**0**	**0**	**0**
Weish9	5	40	MBPSO	0.78	10.61	0.0021	200	25.650
			BPSOTVAC	1	0	0	0	0.000
			CBPSOTVAC	0.85	13.01	0.0027	641	65.700
			QIFAPSO	**0.98**	**0.68**	**0.000129**	**0.34**	**4.784**
Weish10	5	50	MBPSO	0.56	10.84	0.0017	83	22.170
			BPSOTVAC	0.91	1.43	0.0002	68	9.560
			CBPSOTVAC	0.67	57.16	0.0102	1394	188.630
			QIFAPSO	**0.99**	**0.27**	**0.0000425**	**27**	**2.7**

[Chih 2015] and the memetic binary particle swarm optimization (BHTPSO) [Beheshti 2015]. The obtained results globally show that QIFAPSO is more efficient and more robust than these methods, in particular in the case of high diemensional instances of the multi-dimensional knapsack problem.

In the last experiment, we have done a comparison between our algorithm and other PSO-based methods developed recently, namely: BAPSA [Beheshti 2013], BAPSAL [Beheshti 2013] and BPSOL [Liang 2010] on difficult MKP instances

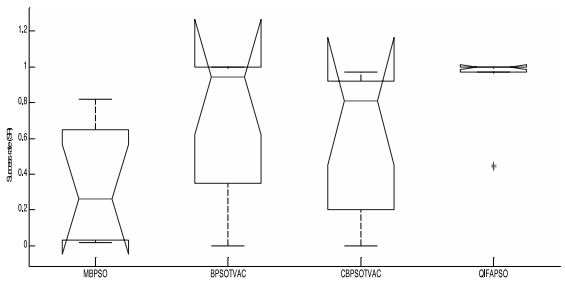


Figure 4.10: Comparing MBPSO, BPSOTVAC, CBPSOTVAC and QIFAPSO wrt anova test for the multidimensional instances sento, and weing.

Table 4.6: Comparison between QIFAPSO, BPSOTVAC and CBPSOTVAC algorithms on high-dimensional knapsack instances (mknapcb3-5-500).

Problem	Algorithm	SR	MAD	MAPE	LE	SD
mknapcb3-5-500-1	BPSOTVAC	0	1414	0.0119	2160	315.67
	CBPSOTVAC	0	1213	0.0102	1906	311.97
	QIFAPSO	**0**	**107.7**	**0.00089639**	**43**	**37.3734**
mknapcb3-5-500-2	BPSOTVAC	0	1304	0.0112	1800	236.66
	CBPSOTVAC	0	1220	0.0105	1693	185.68
	QIFAPSO	**0**	**245.8333**	**0.0021**	**134**	**63.8144**
mknapcb3-5-500-3	BPSOTVAC	0	1332	0.0111	2126	262.2
	CBPSOTVAC	0	1225	0.0102	1928	273.89
	QIFAPSO	**0**	**188.5333**	**0.0016**	**71**	**73.2543**
mknapcb3-5-500-4	BPSOTVAC	0	1418	0.0119	1982	245.5
	CBPSOTVAC	0	1292	0.0108	1730	213.22
	QIFAPSO	**0**	**272.5333**	**0.0023**	**114**	**91.3843**
mknapcb3-5-500-5	BPSOTVAC	0	1400	0.0116	2080	323.44
	CBPSOTVAC	0	1211	0.01	1928	320.16
	QIFAPSO	**0**	**164.2333**	**0.0013**	**80**	**40.1391**
mknapcb3-5-500-6	BPSOTVAC	0	1206	0.0099	1852	232.95
	CBPSOTVAC	0	1065	0.0088	1725	250.14
	QIFAPSO	**0**	**187.2333**	**0.0015**	**63**	**79.218**
mknapcb3-5-500-7	BPSOTVAC	0	1358	0.0115	1799	239.13
	CBPSOTVAC	0	1243	0.0105	1998	281.14
	QIFAPSO	**0**	**142.7**	**0.0012**	**59**	**52.7186**
mknapcb3-5-500-8	BPSOTVAC	0	1164	0.0097	1717	234.55
	CBPSOTVAC	0	1124	0.0094	1944	245.54
	QIFAPSO	**0**	**162.4**	**0.0013**	**68**	**49.892**
mknapcb3-5-500-9	BPSOTVAC	0	1358	0.0113	1775	252.88
	CBPSOTVAC	0	1268	0.0105	1976	248.89
	QIFAPSO	**0**	**462.2333**	**0.0038**	**278**	**130.106**
mknapcb3-5-500-10	BPSOTVAC	0	1266	0.0106	1630	241.13
	CBPSOTVAC	0	1222	0.0102	1824	283.69
	QIFAPSO	**0**	**255.9**	**0.0021**	**156**	**66.226**

whose sizes are ranging from 100 to 500, with a number of constraints between 5 and 30. Table 4.8 presents the best, the average and the worst profit as well as the average error based on 30 independent runs and figure 4.12 depicts the result

Table 4.7: Comparison between QIFAPSO algorithm and SACRO-BPSO-TVAC, SACRO-CBPSO-TVAC and BHTPSO.

Problem	Best known	SACRO-BPSO-TVAC	SACRO-CBPSO-TVAC	BHTPSO	QIFAPSO
mknapcb1-5.100-01	**24381**	24343	24343	24169	**24381**
mknapcb1-5.100-02	**24274**	**24274**	**24274**	24109	**24274**
mknapcb1-5.100-03	**23551**	**23538**	**23538**	23435	**23538**
mknapcb1-5.100-04	**23534**	**23527**	**23527**	23253	23484
mknapcb1-5.100-05	**23991**	**23991**	23966	23815	**23991**
mknapcb1-5.500-01	**120148**	119867	120009	114493	**120105**
mknapcb1-5.500-02	**117879**	117681	117699	112821	**117744**
mknapcb1-5.500-03	**121131**	120951	120923	114774	**121060**
mknapcb1-5.500-04	**120804**	120450	120563	115828	**120690**
mknapcb1-5.500-05	**122319**	122037	122054	115889	**122239**

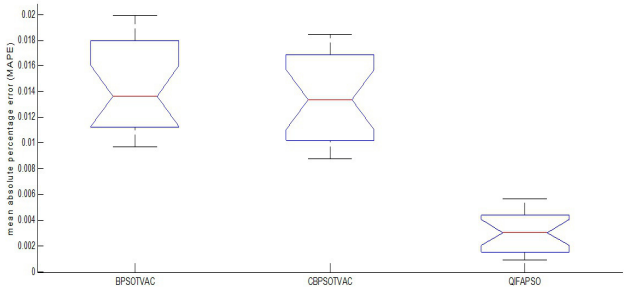


Figure 4.11: Comparing BPSOTVAC, CBPSOTVAC and QIFAPSO wrt anova test for the multidimensional instances mknapcb3-5-500.

of Friedman statistical test. We have noted that the obtained results by our algorithm are better than those obtained by algorithms BAPSA, BAPSAL and BPSOL. Moreover, the result of Friedman statistical test is much better for our algorithm than for the other PSO-based algorithms.

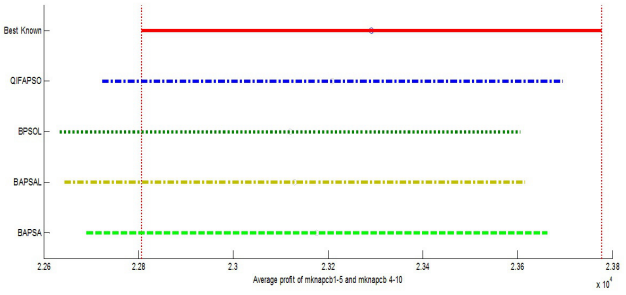


Figure 4.12: Comparing QIFAPSO, BPSOL, BAPSAL and BAPSA with Friedman test for multidimensional knapsack instances of mknapcb1 and mknapcb4.

Table 4.8: Comparison between QIFAPSO algorithm and BAPSA, BPSAL and BPSOL.

Benchmark	Best know	Max profit	BAPSA	BAPSAL	BPSOL	QIFAPSO
mknapcb1-5.100-00	24381	Best	24381	24298	24301	**24381**
		Mean	24243.3	24154.9	24201	**24314**
		Worst	24063	23981	23977	**24146**
mknapcb1-5.100-01	24274	Best	24274	24173	24274	**24274**
		Mean	24102.1	23984.2	23858	**24206**
		Worst	23940	23813	23732	**24116**
mknapcb1-5.100-02	23551	Best	23523	23523	23523	**23538**
		Mean	23503.9	23482.2	23482	**23513**
		Worst	23374	23402	23352	**23444**
mknapcb1-5.100-03	23534	Best	**23486**	23486	23486	23484
		Mean	23370.6	23333	23343.6	**23451**
		Worst	23213	23236	23133	**23354**
mknapcb1-5.100-04	23991	Best	23966	23939	23966	**23991**
		Mean	23897.3	23860.7	23878.5	**23921**
		Worst	23827	23734	23762	**23780**
mknapcb4-10.100-00	23064	Best	23055	23055	23050	**23057**
		Mean	22969.5	**22976.8**	22963	22954
		Worst	22701	22846	22701	**22788**
mknapcb4-10.100-01	22801	Best	**22753**	22739	22706	22750
		Mean	22580.8	22477.4	22459	**22649**
		Worst	22325	22363	22215	**22508**
mknapcb4-10.100-02	22131	Best	22081	22065	22131	**22131**
		Mean	21953.2	21916.5	21901.6	**21977**
		Worst	21786	**21792**	21759	21783
mknapcb4-10.100-03	22772	Best	22643	22587	22569	**22772**
		Mean	22518.9	22482.7	22483.6	**22566**
		Worst	22421	**22355**	22337	22339
mknapcb4-10.100-04	22751	Best	**22751**	22751	22751	22654
		Mean	22621.1	**22627.5**	22624.6	22538
		Worst	22445	**22475**	22447	22380
mknapcb8-30.250-29	150037.662	Best	149121	148915	148975	**149335**
		Mean	148793	148694	148648	**149130**
		Worst	148529	148564	148349	**148875**
mknapcb8-30.500-29	301020.63	Best	298838	298495	297428	**300226**
		Mean	298482	298186	297029	**299940**
		Worst	298320	297950	296443	**299565**
Avg. error of best profit (%)			0.23 %	0.36 %	0.34 %	**0.14 %**
Avg. error of mean profit (%)			0.67 %	0.85 %	0.92 %	**0.50 %**
Avg. error of worst profit (%)			1.29 %	1.33 %	1.59 %	**1.05 %**

4.5 Conclusion and perspectives

In this chapter, we have proposed a new algorithm called QIFAPSO to solve discrete optimization problems. The proposed algorithm is based on the integration into the firefly algorithm of two kinds of concepts: the basic concepts of quantum computation such that the quantum superposition and the quantum measure as well as the PSO formulas of updating position and velocity. The motivation behind this integration is to adapt the firefly algorithm to the case of discrete prob-

lems, to improve its performance in exploring and exploiting the search space and to reduce the population size. We have tested the proposed algorithm on two variants of the knapsack problem: the simple 0-1 knapsack and the multidimensional knapsack problems. In order to validate the feasibility and the efficiency of the proposed algorithm in solving discrete optimization problems, we have based our evaluation on strict and robust criteria including: the average error, in addiction to Anova and Friedman comparison statistical tests. QIFAPSO algorithm has been compared to two classes of algorithms: quantum-inspired algorithms and PSO-inspired algorithms in order to show the relevant and the efficiency of integrating the quantum computation concepts and the PSO style strategy of movement in browsing the search space. The results obtained from QIFAPSO algorithm are very encouraging. Accordingly, we plan to apply it to optimization problems issued from bioinformatics and data mining such as the multiple sequence alignment and the extraction of association rules problems.

CHAPTER 5

Quantum inspired firefly algorithm for feature selection

Contents

5.1 Introduction

In this chapter, we propose an approach for feature selection based on QIFAPSO algorithm developed in [Zouache 2015]. This approach uses the concepts of quantum computing [Manju 2014] including qubit superposition in the representation of the probability of selecting a feature and the quantum measure as a bridge between the quantum representation and the binary representation of selected features (the value 1 indicates that the feature is selected and the value 0 indicates the feature is not selected). In order to improve the search space exploration and find the best subset of features, we propose cooperation between two movement strategies; that of the fireflies swarm [Yang 2009a] and that of PSO particles [Kennedy 1995].

These movements are called quantum movements because they are applied on quantum fireflies represented by sequences of qubit. The impact of these movements on the feature selection appears after the quantum measure of the new

quantum fireflies that result from these movements. In this approach, the evaluation of the feature selection is based on the region position or the dependency degree of the rough set theory and allows one to evaluate the relevance of subsets of condition features (those that are less redundant between each other and more relevant for the decision feature).

5.2 QIFAPSO for feature selection

In this section, we propose a quantum-inspired firefly algorithm based on particle swarm optimization and rough set theory to select the optimal subset of features, i.e., the best reduction of features in an information system. Each subset of features represents a position or a point in the search space. The proposed algorithm explores the search space constituted of 2^N possible subset of attributes where N is the number of input features.

The aim is to find the best subset of features thanks to the synergy between the two following strategies: the movement strategy of fireflies as defined in a firefly algorithm and the particle swarm optimization (PSO) strategy. For that purpose, we put a swarm of fireflies in the space of possible subsets of features. Each firefly takes a position. The fireflies swarm explores this space to search the best position, i.e., arriving to a firefly with the most possible brightness). During the search process, the fireflies change their positions by applying either the FA movement strategy where a less bright firefly moves towards a brighter firefly or the PSO movement where a firefly changes its position by taking into account its best position up to the current iteration together with the global best position in the whole firefly swarm. Our approach uses the concepts of quantum computing [Manju 2014], namely qubit superposition in the representation of the probability of selecting features, and the quantum measure as a bridge between quantum representation and binary representation of a feature selection (the value 1 indicates the selection of the feature and the value 0 indicates its rejection). The evaluation of feature selection is based on the position region or the dependency degree concepts issued from rough set theory and allows one to evaluate the relevance of subsets of condition features (those that are less redundant between each other and more relevant for the decision feature).

5.2.1 Quantum representation for feature selection

Fireflies' positions are represented by a sequence of quantum bits of length N where N is the total number of condition features: $N = card(C)$. Each qubit en-

Algorithm 5.1 The pseudo-code of QIFAPSO-FS

Input: An information table and the parameters of QIFAPSO algorithm
Input: A minimal feature reduct R_{min} and its cardinality L_{min}
Output: $R_{min} \leftarrow C$;
Output: $L_{min} \leftarrow |C|$;

```
iteration ← 0;
while (iterationmaxcycle) do
    for each fireflie i do
        for each fireflie j do
            if I(Bf_i)I(Bf_j) then
                R_i ← updating_mouvment of FA()
            else
                R_i ← updating_mouvment of PSO()
            end if
        end for
    end for
end while
```

codes the probability of selecting or not a feature. A qubit is represented by the couple $(\cos(\theta), \sin(\theta))$ of a rotation angle θ and is given by the following formula:

$$|Q>=\cos(\theta)|0>+\sin(\theta)|1> \quad \textit{such that } |\cos(\theta)|^2+|\sin(\theta)|^2=1 \tag{5.1}$$

$$\begin{cases} |\cos(\theta)|^2 & \textit{represent the probability of the not selecting a feature c} \\ |\sin(\theta)|^2 & \textit{represent the probability of the selecting a feature c} \end{cases}$$

Accordingly, a position of a quantum firefly encodes indeed a state superposition of several binary solutions, i.e., several possible subsets of features. Thus, the quantum representation ensures more diversity in the population of solutions and does not require a great number of fireflies to find the optimal subset of features in the search space.

5.2.2 Construction of feasible solution by Quantum observation

In QIFAPSO, we use the quantum measure operation to construct a binary solution which corresponds to an effective selection of a subset of features. Each bit of the binary solution is created by projecting a qubit of the quantum solution. For each qubit, we generate a random number r in the interval $[0,1]$. Then, the corresponding bit of the binary solution takes the value 1, i.e., the corresponding feature is selected if $r > |\sin(\theta)|^2$, otherwise it takes the value 0 and the corresponding feature is not selected.

Algorithm 5.2 Quantum observation

Input: quantum firefly Qf, $C = \{c_1, c_2, \cdots, c_n\}$the set of features
Output: binary solution Bf, subset of features R

```
R ← Φ
for each qubit i of Qf do
    Generate random real value r between 0 and 1
    if (r > cos(θ_i)^2) then
        Bf_i ← 1
        R ← R ∪ c_i
    else
        Bf_i ← 0
    end if
end for
```

5.2.3 Fitness function

To evaluate the relevance of a subset R of features of the reduction of the set of condition features with respect to the decision feature D in an incomplete information system, we propose to use the evaluation function defined by Xiangyang Wang in [Wang 2007c] as follows:

$$fitness = \alpha \times \gamma_R(D) + \beta \times \frac{(|C| - |R|)}{|C|} \tag{5.2}$$

where $\gamma_R(D)$ is the dependency degree of the subset R of condition features with respect to the decision feature D. $|R|$ is the cardinality of the set R of selected features with $R \subset C$, i.e., the number of 1 in a binary firefly solution. $|C|$ is the total number of condition features. α and β are two coefficients that correspond to the impact of the dependency degree and the number of selected features.

From the optimization point of view, QIFAPSO algorithm tries to maximize the fitness function or to find an optimal solution (the best subset of features) having a maximal value of the evaluation function, i.e., having a maximal dependency degree and a minimal number of features. Thus, in the best solution, the selected condition features are those that have the maximal influence on the decision feature D while minimizing the redundancy of the selected features between each other.

5.2.4 The distance and attractiveness between two fireflies' solutions

Let Bf_i, Bf_j be two binary solutions corresponding to two fireflies. We use the variant of Hamming distance proposed in [Hamming 1950]. This distance is the proportion of the number of different positions between two binary solutions (the

number of 1 from Bf_i XOR Bf_j) and the number of zeros in these two binary solutions.

$$Distance(Bf_i, Bf_j) = \frac{\sum_{k=1}^{N}(Bf_{ik} \oplus Bf_{jk})}{2 \times N - (\sum_{k=1}^{N} Bf_{ik} + \sum_{k=1}^{N} Bf_{jk})} \tag{5.3}$$

where $\oplus$ is the addition modulo 2, $\exists r_1 \wedge r_2 \in \{1, \cdots, N\} | Bf_{ir1} \neq 0$ *and* $Bf_{ir2} \neq 0$,$0 \leq Distance(Bf_i, Bf_j) \leq 1$.

The distance between two fireflies increases when the number of different positions in the two fireflies increases or when the number of identical positions containing the value 1 between the two fireflies increases, and vice versa. The attractiveness between two binary fireflies is given by the following formula:

$$Att(Bf_i, Bf_j) = e^{-\gamma \times Distance(Bf_i, Bf_j)^2} \tag{5.4}$$

where γ is a parameter indicating the environment absorption. The attractiveness between two binary fireflies increases when the distance between the two fireflies decreases and vice versa. Consequently, the higher is the attractiveness between two fireflies Bf_i and Bf_j , the more is the importance of the movement of Bf_i towards Bf_j (supposing that Bf_i is less bright than Bf_j) and vice versa. The meaning and the importance of this distance in exploring the search space to find the best subset of features is that if a subset of features R_i corresponding to a firefly bf_i is less performing than another subset of features R_j corresponding to a firefly bf_j. (according to the objective function defined in section 5.3.3).

The following example illustrates the efficiency of this distance. Let $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ be the set of condition features. Let us consider two feature selection configurations for two binary fireflies Bf_i and Bf_j. In the first configuration, the value of the binary firefly Bf_i is $(0,1,0,1,1,1)$ which corresponds to the subset of features $R_i = \{c_2, c_4, c_5, c_6\}$ and the value of the binary firefly Bf_j is $(1,0,1,1,1,1)$ which corresponds to the subset of features $R_j = \{c_1, c_3, c_4, c_5, c_6\}$. In the second configuration, the value of the binary firefly Bf_i is $(0,1,0,0,0)$ which corresponds to the subset of features $R_i = \{c_2\}$ and the value of the binary firefly Bf_j is $(1,0,1,0,0,0)$ which corresponds to the subset of features $R_j = \{c_1, c_3\}$. The distance between the two fireflies Bf_i and Bf_j in the first configuration equals 1 whereas it equals 0.33 in the second configuration. The attractiveness between the two fireflies in the first configuration is smaller than that in the second configuration.

In summary, this distance favors solutions with less number of features and avoids the premature convergence problem since the firefly is more attracted by another firefly and moves towards it in the case where the number of identical

attributes that are not selected by both fireflies is greater (this intuition gives more freedom in the exploration of the search space by the swarm of fireflies and ensures the preservation of the population diversity).

5.2.5 Quantum movements for updating the fireflies' solutions

The exploration of the search space of subsets of features is done thanks to the cooperation of two movement strategies. The first strategy is inspired from the movement of firefly swarms and the second one is inspired from the PSO movement.

The first strategy is applied in the case where a binary firefly Bf_i is less bright than another binary firefly Bf_j (i.e., the subset of selected features corresponding to the firefly Bf_i is less relevant than the subset of selected features corresponding to the firefly Bf_j). In this strategy, the binary firefly Bf_i is attracted by Bf_j and moves towards it. The effect of this movement appears in the change of the subset of selected features corresponding to binary firefly Bf_i as presented in the algorithm 5.3 .

The PSO movement strategy is applied in the case where the condition for the application of the firefly movement strategy is not satisfied i.e., in the case where the subset of features corresponding to firefly Bf_i is more relevant than the subset of features corresponding to firefly Bf_j according to the objective function (the dependency degree and the number of selected features). The firefly Bf_i moves to another position by taking into account its best position in its history and the current in the swarm fireflies (see the algorithm 5.4).

The two movement strategies are applied on the set of quantum fireflies. These movements allow increasing or decreasing the probability to observe the value 1 from the quantum state of each qubit of a quantum firefly. The effect of the quantum movement on the feature selection is to change the probability of selecting a feature. This effect in concretized on the set of binary solutions (the set of subsets of attributes) after the application of the quantum measure on each qubit of a quantum firefly.

5.3 Experimental results

In this section, the performances of our feature selection approach are proved. The objective is to evaluate the QIFAPSO algorithm in terms of number of features and accuracy of set classification of selected features. QIFAPSO-FS is run on twelve discrete UCI datasets extracted from the repository of machine learning databases

Algorithm 5.3 Updating quantum firefly solution by FA movement

Input: Θ_i^t, Θ_j^t: two quantum fireflies i and j at iteration t,
Input: Bf_i^t, Bf_j^t:two binary fireflies i and j at iteration t,
Input: C :the set of condition features.
Output: Θ_i^{t+1}: the new quantum firefly i at iteration $t+1$,
Output: Bf_i^{t+1}: the new binary firefly i at iteration $t+1$,
Output: R_i^{t+1}:the new subset of features corresponding to binary firefly Bf_i^{t+1}.

$R_i^{t+1} \leftarrow \Phi$;
Computing the distance and attractiveness between the two current binary fireflies Bf_i^t and Bf_j^t;
for each bit k of binary firefly Bf_i^t **do**
  if $Bf_{ik}^t \neq Bf_{jk}^t$ **then**
    $\Theta_{ik}^{t+1} \leftarrow \Theta_{ik}^t + Attr(Bf_i^t, Bf_j^t)(\Theta_{ik}^t - \Theta_{jk}^t) + \alpha \times rand()$;
  else
    $\Theta_{ik}^{t+1} \leftarrow \Theta_{ik}^t$;
  end if
  if $rand() > cos(\Theta_{ik}^{t+1}))^2$ **then**
    $Bf_{ik}^{t+1} \leftarrow 1$
    $\mathrm{R}_i^{t+1} \leftarrow R_i^{t+1} \cup c_k$
  else
    $Bf_{ik}^{t+1} \leftarrow 0$
  end if
end for

[UCI 1998]. These datasets is presented in table 5.1. The column 'feature' and the column 'instance' give respectively the total number of features and instances in a dataset. The column 'accuracy_Tree' and the column 'accuracy_KNN' present respectively the classification accuracy given by the decision tree algorithm and the classification algorithm in an original dataset containing the complete set of features.

The evaluation of the performance of the feature selection based on the proposed algorithm in the classification task is proved by the cross validation method. This method decomposes the training set into n samples and each sample constructs a classification model or a classifier according to the used classification algorithm. Moreover, each model is tested and evaluated $(n-1)$ times on its classification accuracy by using the other $(n-1)$ samples. The accuracy rates of the training set are given by the average of classification accuracy rates of the set of models generated by the n samples. This method is more robust and more efficient in the evaluation of the classification accuracy. More precisely, in our case, we use the 10-fold cross validation method where the training set is decomposed into 10 samples. To generate the classification model for each sample, the two following

Algorithm 5.4 Updating quantum firefly solution by PSO movement

Input: Θ_i^t, Θl_i^t: the quantum firefly i and its best quantum until iteration t, Θg^t the best quantum position of the set of fireflies at iteration t,
Input: Bf_i^t, Bfl_i^t: the binary firefly i and its best binary position until iteration t, Bfg^t the best binary position of the set of binary fireflies at iteration t,
Input: $\Delta\theta_i^t$: Vector of rotation angles, initialized to 0 at iteration $t = 0$.
Input: C: the set of condition features.
Output: Θ_i^{t+1}: the new quantum firefly i at iteration $t+1$,
Output: Bf_i^{t+1}:the new binary firefly i at iteration $t+1$.
Output: R_i^{t+1}: the new subset of features corresponding to binary firefly Bf_i^{t+1}.

$\Delta\theta_i^{t+1}$: the new vector of rotation angles at iteration $t+1$.
$R_i^{t+1} \leftarrow \Phi$;
for each bit k of binary firefly Bf_i^t **do**
  if $(Bf_i^t \neq Bfl_i^t) \vee (Bf_i^t \neq Bfg^t)$ **then**
    $\Delta\theta_{ik}^{t+1} \leftarrow w.\Delta\theta_{ik}^t + rand() \times C_1 \times (\Theta_{ik}^t - \Theta g_k^t) + rand() \times C_2 \times (\Theta_{ik}^t - \Theta l_{ik}^t)$;
    $\Theta_{ik}^{t+1} \leftarrow \Theta_{ik}^{t+1} + \Theta_{ik}^t$;
  end if
  if $rand() > cos(\Theta_{ik}^{t+1}))^2$ **then**
    $Bf_{ik}^{t+1} \leftarrow 1$
    $R_i^{t+1} \leftarrow R_i^{t+1} \cup c_k$
  else
    $Bf_{ik}^{t+1} \leftarrow 0$
  end if
end for

Table 5.1: Data description.

	Feature	Instances	Accuracy_Tree	Accuracy_KNN
Lymphography	18	148	0.79 ± 0.08	0.79 ± 0.10
Mushroom	22	8124	1 ± 0	1 ± 0
Soybean-small	35	47	0.97 ± 0.06	1 ± 0
Zoo	16	101	0.87 ± 0.08	0.96 ± 0.05
Dermatology	34	366	0.93 ± 0.03	0.94 ± 0.05
Vote	16	300	0.93 ± 0.04	0.91 ± 0.04
Lung	56	32	0.84 ± 0.16	0.65 ± 0.19
Breast-cancer	9	699	0.93 ± 0.02	0.95 ± 0.02
DNA	57	318	0.36 ± 0.13	0.4811 ± 0.05
Exactly	13	1000	0.75 ± 0.07	0.74 ± 0.0437
Exactly2	13	1000	0.68 ± 0.04	0.66 ± 0.02
Led	24	2000	1 ± 0	0.78 ± 0.01
Average	26.083		0.84 ± 0.06	0.82 ± 0.05

classification algorithms are used: decision tree algorithm and KNN algorithm.

The experimental results of the proposed algorithm applied on twelve datasets are summed up in table 5.2. In this table, the performances of QIFAPSO-FS algorithm are presented in terms of the number of reduced features, the reduction rate

of the total number features, the accuracy of the classification of the feature subsets obtained by QIFAPSO-FS algorithm and finally, the optimal solution i.e., the optimal subset of features.

Table 5.2: Performance comparison on discrete datasets.

	Length of the reducts	Number of reducts	Rate of reducts	Best Accuracy Tree	Best Accuracy KNN	Feature selection
Lymphography	7	11	61.11	0.79 ± 0.10	0.80 ± 0.10	2, 6, 8, 13, 14, 15, 18
Mushroom	4	18	81.81	0.99 ± 0.01	0.99 ± 0.01	5, 12, 20, 22
Soybean-small	2	33	94.28	1 ± 0	1 ± 0	22, 23
Zoo	5	11	68.75	0.94 ± 0.05	0.94 ± 0.05	3, 4, 6, 8, 13
Dermatology	10	24	70.58	0.87 ± 0.04	0.78 ± 0.04	1, 5, 6, 9, 16, 19, 21, 26, 28, 32
Vote	8	8	50	0.93 ± 0.05	0.92 ± 0.05	1, 2, 3, 4, 9, 11, 13, 16
Lung	4	52	92.85	0.84 ± 0.17	0.84 ± 0.17	3, 9, 24, 42
Breast-cancer	4	5	55.55	0.94 ± 0.01	0.95 ± 0.01	1, 5, 6, 8
DNA	5	52	91.22	0.36 ± 0.09	0.33 ± 0.05	5, 19, 22, 26, 33
Exactly	6	7	53.84	1 ± 0	1 ± 0	1, 3, 5, 7, 9, 11
Exactly2	10	3	23.07	0.72 ± 0.02	0.79 ± 0.02	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Led	5	19	79.16	1 ± 0	1 ± 0	1, 2, 3, 4, 5
Average	5.81			0.868 ± 0.04	0.86 ± 0.04	

According to the obtained results presented in table 5.2, figures 5.1-5.3, we observe that the proposed algorithm is more efficient in reducing of the number of features. For instance, in the datasets Soybean-small and DNA, the reduction rate is more than 90%. The rate of classification accuracy in feature selection obtained by the decision tree or the KNN algorithms is almost equal to the rate of classification accuracy of the complete set of features, in most datasets. In some cases, the classification accuracy of the subset of selected features by QIFAPSO-FS algorithm is better compared with the complete set of features as it is the case for instance for Soybean-small, Exactly2 and Led datasets.

The process of search space exploration by QIFAPSO-FS algorithm for the following datasets: vote, mushroom, exactly2, DNA and dermatology are presented in tables 5.3-5.7 respectively and in figures 5.4-5.8 respectively. In each table, the 'best solution' presents the best subset of features found by QIFAPSO-FS algorithm at each iteration. The 'fitness value' denotes the evaluation criteria for the relevance of a subset of features at each iteration. The fitness value combines two

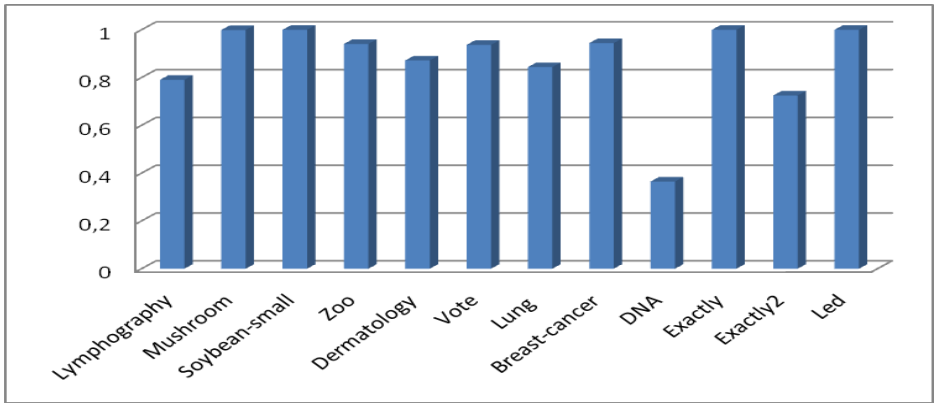


Figure 5.1: Reduction rate given by QIFAPSO-FS for all datasets.

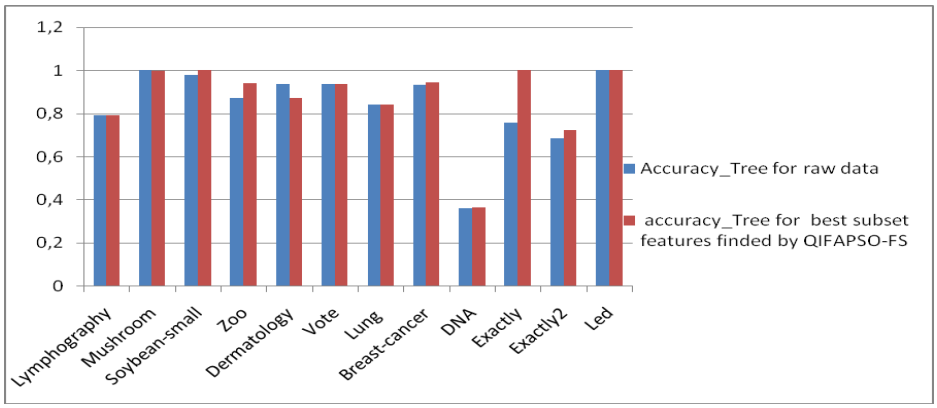


Figure 5.2: Accuracy classification given by tree decision for raw data and best subset find by QIFAPSO-FS.

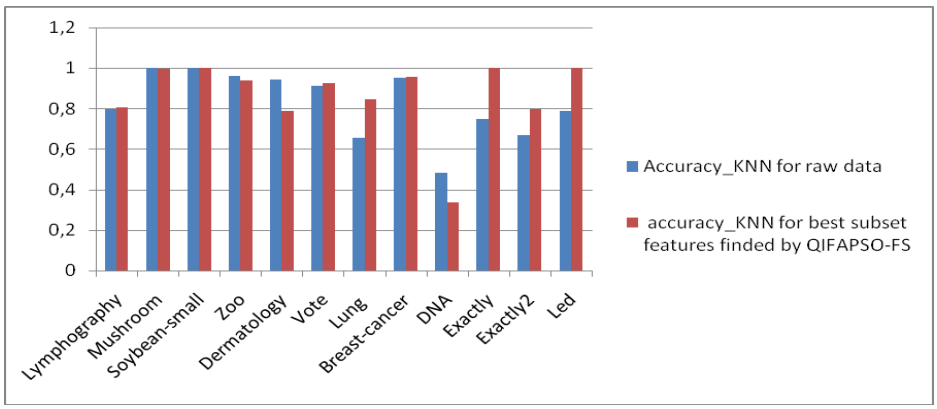


Figure 5.3: Accuracy classifications given by KNN algorithm for raw data and best subset find by QIFAPSO-FS.

evaluation criteria in order to assess a subset of features: The first one presents the dependency degree (of rough set theory) with the decision feature; the second criterion gives the number of reduced features. In our case, the coefficients of the two criteria are fixed as follows: $\alpha = 0.9$ and $\beta = 0.1$, and the maximal number of iterations is 20.

Table 5.3: QIFAPSO-FS exploration process on vote.

Iter	Best solution	Fitness value	Feature subset length
1	3, 4, 5, 6, 7, 10, 11, 13, 15	0.9108	9
2	2, 3, 4, 7, 10, 11, 12, 13, 16	0.9287	9
3	3, 4, 7, 9, 10, 11, 12, 13, 16	0.9317	9
4–5	2, 3, 4, 11, 12, 13, 16	0.9413	7
6–7	1, 2, 3, 4, 11, 12, 13, 16	0.944	8
8	1, 2, 3, 4, 9, 11, 13, 16	0.95	8

Table 5.4: QIFAPSO-FS exploration process on mushroom.

Iter	Best solution	Fitness value	Feature subset length
1	1, 3, 4, 5, 10, 11, 20	0.9682	7
2	3, 5, 11, 12, 16, 22	0.9727	6
3	5, 11, 12, 19, 22	0.9773	5
4	5, 12, 20, 22	0.9783	4

Table 5.5: QIFAPSO-FS exploration process on exactly2.

Iter	Best solution	Fitness value	Feature subset length
1–2	1, 2, 4, 6, 7, 8, 9, 11, 12, 13	0.7206	10
3	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13	0.8636	12
4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12	0.9154	11
5	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0.9231	10

Table 5.6: QIFAPSO-FS exploration process on DNA.

Iter	Best solution	Fitness value	Feature subset length
1	3, 4, 5, 11, 12, 14, 16, 18, 21, 33, 36, 37, 39, 42, 46, 48, 49, 50, 53, 57	0.9649	15
2–3	2, 3, 9, 10, 44, 52, 55	0.9877	7
4	4, 20, 22, 33, 43, 44	0.9895	6

According to tables 5.3-5.7 and figures 5.4-5.8, we can see that QIFAPSO-FS algorithm is more efficient in the exploration and the exploitation of the search space of the possible subsets of features thanks to the diversity of the set of solutions provided by the concepts of quantum computation such as the superposition of qubit states and quantum measure as well as the cooperation between the two movement strategies issued from PSO and FA.

Table 5.7: QIFAPSO-FS exploration process on dermatology.

Iter	Best solution	Fitness value	Feature subset length
1	1, 2, 3, 7, 8, 9, 10, 11, 16, 17, 19, 20, 16, 24, 28 ,30 ,31	0.9431	17
2–3	1, 5, 8, 15, 16, 18, 19, 20, 21, 22, 28, 29, 32	0.9519	13
4	1, 3, 5, 10, 14, 15, 16, 18, 19, 22, 25, 26, 14, 28, 32	0.9588	15
5	1, 2, 3, 5, 8, 14, 15, 19, 21, 25, 26, 28, 30	0.9618	13
6	1, 2, 5, 6, 9, 12, 15, 16, 18, 19, 25, 26	0.9647	12
7	1, 3, 5, 14, 15, 16, 19, 21, 26, 28, 32	0.9676	11
8	1, 5, 6, 9, 16, 19, 21, 26, 28, 32	0.9706	10

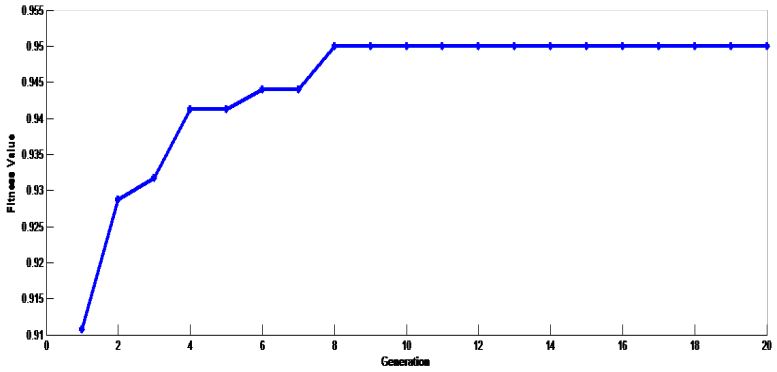


Figure 5.4: Evolution exploration process of the global best on dataset vote by QIFAPSO-FS algorithm.

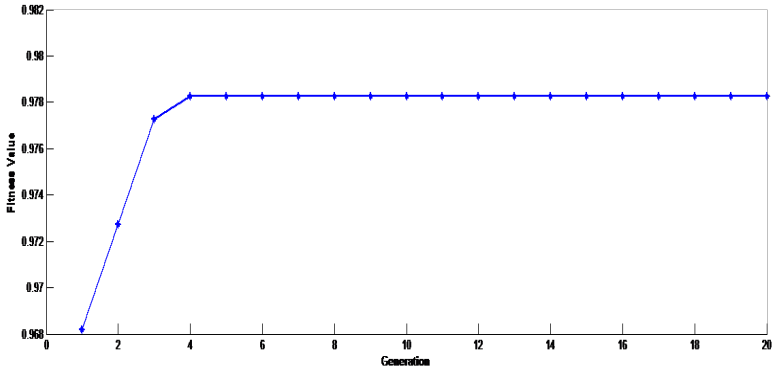


Figure 5.5: Evolution exploration process of the global best on dataset mushroom by QIFAPSO-FS algorithm.

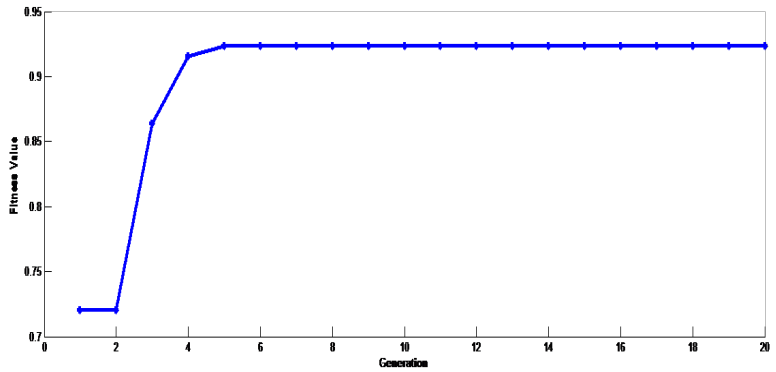


Figure 5.6: Evolution exploration process of the global best on dataset exactly2 by QIFAPSO-FS algorithm.

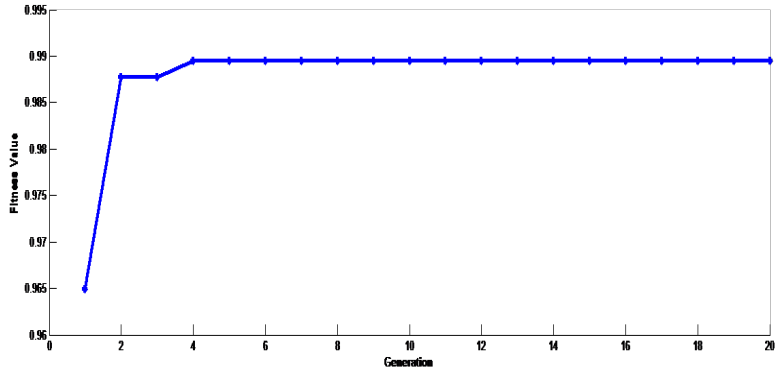


Figure 5.7: Evolution exploration process of the global best on dataset DNA by QIFAPSO-FS algorithm.

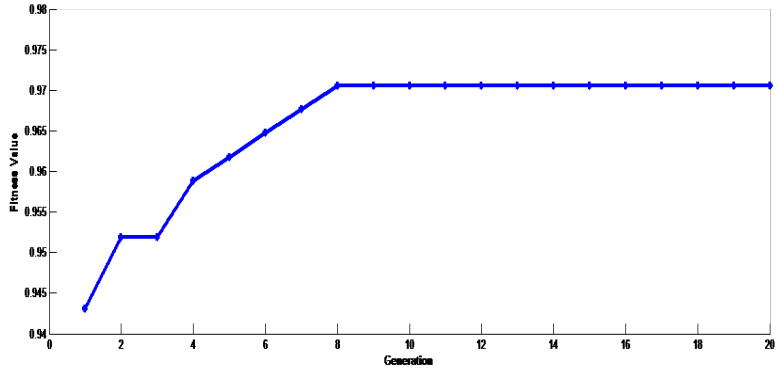


Figure 5.8: Evolution exploration process of the global best on dataset dermatology by QIFAPSO-FS algorithm.

The results of the comparison of QIFAPSO-FS algorithm with the algorithms of feature selection based on rough set theory are presented in table 8. These algorithms are: PSORSFS [Wang 2007c] , FSARSR[Chen 2015]. According to the number of reduced features criteria, we observe that the our algorithm is more competitive than the four considered algorithms in all the datasets used in this comparison. In the case of 'vote' dataset, QIFAPSO-FS gives a better feature selection than these algorithms.

Table 5.8: Reduct sizes found by feature selection algorithms.

Dataset	features	PSORSFS	FSARSR	QIFAPSOFS
Lung	56	**4**	**4**	**4**
Mushroom	22	**4**	**4**	**4**
Soybon small	35	**2**	**2**	**2**
Vote	16	**8**	9	**8**

Another comparison is done with respect to the minimal number of iterations to find the best solution. In this comparison, QIFAPSO-FS algorithm is compared with the two following metaheuristics: Particle swarm optimization based on rough set for feature selection (PSORSFS) [Wang 2007c] and finding rough set reducts with fish swarm algorithm (FSARSR) [Chen 2015]. The results of this comparison are given in table 5.9 and show that for all datasets used in this comparison, QIFAPSO-FS algorithm finds the best solution after a number of iterations which is smaller than that for the two algorithms PSORSFS and FSARSR. These results prove that QIFAPSO-FS algorithm is more efficient in the exploration and the exploitation of the search space.

Table 5.9: Minimal number of iterations to find the best subset of features by metaheuristic algorithms.

Dataset	PSORSFS	FSARSR	QIFAPSOFS
Lung	40	10	5
Mushroom	16	7	4
Soybon small	33	10	4
Vote	54	8	5

5.4 Chapter summary

In this chapter, we have applied quantum inspired firefly with particle swarm optimization algorithm presented in the previous chapter for the feature selection task. This task is crucial in improving the quality of grouping obtained by clustering algorithms. Also, it increases the performance of classification algorithms. Furthermore, feature selection allows to reduce the grouping complexity and to construct

the classification model. In this work, we have chosed to apply QIFAPSO-FS algorithm for feature selection in order to improve the performance of classification. We have used concepts form rough set theory to evaluate the relevance of the subsets of features found during the process of search space exploration performed by QIFAPSO-FS algorithm. The performances of QIFAPSO-FS algorithm are evaluated through the measure of the classification accuracy given by the cross validation method and the features reduction rate. The obtained experimental results showed that the proposed algorithm outperforms the algorithms we have used for the comparison both in terms of classification accuracy and in terms of reduction rate.

CHAPTER 6

Conclusions and Future works

Contents

6.1 Conclusions

The focal point of this thesis is the development of bio-inspired algorithms to solve data mining problems. We have exploited two main resources in developing two proposed algorithms, namely QIFAPSO and QDEPSO: The first resource is the collective intelligence of the stack of swarm intelligence algorithms and the second resource is the concepts of quantum computing. We have adopted a strategy which consists in developing bio-inspired algorithms to solve general optimization problems before instantiating them for specific data mining problems.

The first proposed algorithm is called QDEPSO and is intended to solve global optimization discrete problems. QDEPSO is a hybridization of three meta- heuristic methods, namely: Differential Evolution Algorithm, Particle Swarm Optimization and Quantum-Inspired Evolutionary Algorithm. QDEPSO uses the concepts of quantum computing such as the superposition state of the qubit and the quantum gate. Moreover, it uses the differentiation operations of the differential evolution algorithm such as mutation, crossover and selection. To reach a good tradeoff between exploration and exploitation of the search space of solutions, the algorithm uses adapts update formulas of the PSO in updating the population of solutions.

In this thesis, we have also proposed a new algorithm called QIFAPSO to solve discrete optimization problems. The proposed algorithm is based on the integration into the firefly algorithm of two kinds of concepts: the basic concepts of quantum computation such that the quantum superposition and the quantum measure as well as the PSO formulas of updating position and velocity. The motivation

behind this integration is to adapt the firefly algorithm to the case of discrete problems, to improve its performance in exploring and exploiting the search space and to reduce the population size.

In this thesis, we have chosed to apply QIFAPSO-FS algorithm for feature selection in order to improve the performance of classification. We have used concepts form rough set theory to evaluate the relevance of the subsets of features found during the process of search space exploration performed by QIFAPSO-FS algorithm.

6.2 Future works

I believe that the objectives intended by this thesis have been reached. These objectives can be summed up as follows:

- Discovering the domain of knowledge extraction: its methods and its problems;
- Discovering the domain of bio-inspired algorithms: its philosophy in inspiring algorithmic methods from nature, emergence and collective intelligence and their use in modeling complex systems and solving difficult problems within these systems;
- Integrating the new discipline of computer science which is based on quantum physics as a main component in our approach of solving very difficult optimization problems;
- Developing effective algorithms based on the domains evoked below to solve optimization problems and adapting them to data mining applications.

Understanding in depth these research lines and propose numerous and substantial contributions in this field remains my long term objective. This thesis is a first step in this investigation process which will need further efforts and years of training and work. My short-term objectives consist in completing the works initiated in the present thesis, namely:

- Inspiring new algorithms from nature in general and from animals and human beings life in particular, to solve optimization problems;
- Applying the two proposed algorithms QIFAPSO and QDEPSO in other problems from data mining such as association rules and clustering among others;

- Using the new quantum gates and other quantum phenomena that have not yet been used in the literature to solve optimization problems.

Bibliography

[Akama 2015] Akama and Seiki. Elements of quantum computing history, theories and engineering applications. Springer, New York, 2015. (Cited on pages 4 and 10.)

[Amaldi 1998] Edoardo Amaldi and Viggo Kann. *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*. Theoretical Computer Science, vol. 209, no. 1, pages 237–260, 1998. (Cited on page 10.)

[B 2014] Jiang B and N. Wang. *Cooperative bare-bone particle swarm optimization for data clustering*. Soft Comput, vol. 18, no. 6, page 1079–1091, 2014. (Cited on page 3.)

[Banati 2011] Hema Banati and Monika Bajaj. *Fire fly based feature selection approach*. IJCSI International Journal of Computer Science Issues, vol. 8, no. 4, 2011. (Cited on page 4.)

[Bansal 2012] Jagdish Chand Bansal and Kusum Deep. *A modified binary particle swarm optimization for knapsack problems*. Applied Mathematics and Computation, vol. 218, no. 22, pages 11042–11061, 2012. (Cited on pages 45 and 59.)

[Battiti 1994] Roberto Battiti. *Using mutual information for selecting features in supervised neural net learning*. Neural Networks, IEEE Transactions on, vol. 5, no. 4, pages 537–550, 1994. (Cited on page 21.)

[Beasley 2005] JE. Beasley. *ORLib operations research library*. `http://people.brunel.ac.uk/_mastjjb/jeb/orlib/mknapinfo.html`, 2005. (Cited on page 57.)

[Beheshti 2013] Zahra Beheshti, Siti Mariyam Shamsuddin and Siti Sophiayati Yuhaniz. *Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems*. Journal of Global optimization, vol. 57, no. 2, pages 549–573, 2013. (Cited on pages 45, 58 and 62.)

[Beheshti 2015] Zahra Beheshti, Siti Mariyam Shamsuddin and Shafaatunnur Hasan. *Memetic binary particle swarm optimization for discrete optimization problems*. Information Sciences, vol. 299, pages 58–84, 2015. (Cited on page 62.)

[Benioff 1980] Paul Benioff. *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines*. Journal of Statistical Physics, vol. 22, no. 5, pages 563–591, 1980. (Cited on pages 4, 10 and 25.)

[Blum 1997] Avrim L Blum and Pat Langley. *Selection of relevant features and examples in machine learning*. Artificial intelligence, vol. 97, no. 1, pages 245–271, 1997. (Cited on page 15.)

[Brown 2012] Gavin Brown, Adam Pocock, Ming-Jie Zhao and Mikel Luján. *Conditional likelihood maximisation: a unifying framework for information theoretic feature selection*. The Journal of Machine Learning Research, vol. 13, no. 1, pages 27–66, 2012. (Cited on page 21.)

[Caruana 1994] Rich Caruana and Dayne Freitag. *Greedy Attribute Selection.* In ICML, pages 28–36. Citeseer, 1994. (Cited on page 23.)

[Chang 2010] Chih-Cheng Chang, Chi-Yuan Chen, Cheng-Wei Fan, Han-Chieh Chao and Yao-Hsin Chou. *Quantum-inspired electromagnetism-like mechanism for solving 0/1 knapsack problem*. In Information Technology Convergence and Services (ITCS), 2010 2nd International Conference on, pages 1–6. IEEE, 2010. (Cited on pages 26 and 31.)

[Chatterjee 2012] Anirban Chatterjee, Gautam Kumar Mahanti and Arindam Chatterjee. *Design of a fully digital controlled reconfigurable switched beam concentric ring array antenna using firefly and particle swarm optimization algorithm*. Progress In Electromagnetics Research B, vol. 36, pages 113–131, 2012. (Cited on page 4.)

[Chen 2010] Yumin Chen, Duoqian Miao and Ruizhi Wang. *A rough set approach to feature selection based on ant colony optimization*. Pattern Recognition Letters, vol. 31, no. 3, pages 226–233, 2010. (Cited on page 23.)

[Chen 2014] Chun-Lung Chen, Shin-Ying Huang, Yeu-Ruey Tzeng and Chuen-Lung Chen. *A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem.* Soft Computing, vol. 18, no. 11, pages 2271–2282, 2014. (Cited on page 3.)

[Chen 2015] Yumin Chen, Qingxin Zhu and Huarong Xu. *Finding rough set reducts with fish swarm algorithm*. Knowledge-Based Systems, vol. 81, pages 22–29, 2015. (Cited on pages 23 and 80.)

[Chiang 2014] Hua-Pei Chiang, Yao-Hsin Chou, Chia-Hui Chiu, Shu-Yu Kuo and Yueh-Min Huang. *A quantum-inspired Tabu search algorithm for solving combinatorial optimization problems*. Soft Computing, vol. 18, no. 9, pages 1771–1781, 2014. (Cited on page 5.)

[Chih 2014] Mingchang Chih, Chin-Jung Lin, Maw-Sheng Chern and Tsung-Yin Ou. *Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem*. Applied Mathematical Modelling, vol. 38, no. 4, pages 1338–1350, 2014. (Cited on pages 45 and 59.)

[Chih 2015] Mingchang Chih. *Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem*. Applied Soft Computing, vol. 26, pages 378–389, 2015. (Cited on page 62.)

[Chou 2011] YH. Chou, YJ. Yang and CH. Chiu. *Classical and quantum-inspired Tabu search for solving 0/1 knapsack problem*. In Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on, pages 1364–1369. IEEE, 2011. (Cited on page 26.)

[Chu 1998] Paul C Chu and John E Beasley. *A genetic algorithm for the multidimensional knapsack problem*. Journal of heuristics, vol. 4, no. 1, pages 63–86, 1998. (Cited on page 45.)

[Congying 2011] Lv Congying, Zhao Huanping and Yang Xinfeng. *Particle swarm optimization algorithm for quadratic assignment problem*. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on, volume 3, pages 1728–1731. IEEE, 2011. (Cited on page 3.)

[Cover 1977] Thomas M Cover and Jan M Van Campenhout. *On the possible orderings in the measurement selection problem*. Systems, Man and Cybernetics, IEEE Transactions on, vol. 7, no. 9, pages 657–661, 1977. (Cited on page 10.)

[Cover 2012] Thomas M Cover and Joy A Thomas. Elements of information theory. John Wiley & Sons, 2012. (Cited on page 18.)

[Dahl 2009] David B Dahl*et al.* *Modal clustering in a class of product partition models*. Bayesian Analysis, vol. 4, no. 2, pages 243–264, 2009. (Cited on page 2.)

[Dash 1997] Manoranjan Dash and Huan Liu. *Feature selection for classification*. Intelligent data analysis, vol. 1, no. 1, pages 131–156, 1997. (Cited on pages ix, 16, 17 and 22.)

[Degang 2007] Chen Degang, Wang Changzhong and Hu Qinghua. *A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets*. Information Sciences, vol. 177, no. 17, pages 3500–3518, 2007. (Cited on page 23.)

[Dempster 1967] Arthur P Dempster. *Upper and lower probabilities induced by a multivalued mapping*. The annals of mathematical statistics, pages 325–339, 1967. (Cited on page 22.)

[Desmedt 2011] Yvo Desmedt. *Knapsack cryptographic schemes*. In Encyclopedia of Cryptography and Security, pages 695–704. Springer, 2011. (Cited on page 44.)

[Dong 2009] Jiwen Dong and Ruihai Wu. *Diversity guided immune clonal quantum-behaved particle swarm optimization algorithm and the wavelet in the forecasting of foundation settlement*. In Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on, pages 3–573. IEEE, 2009. (Cited on page 26.)

[Dorigo 1999] Marco Dorigo, Gianni Di Caro and Luca Maria Gambardella. *Ant algorithms for discrete optimization*. Artificial life, vol. 5, no. 2, pages 137–172, 1999. (Cited on page 3.)

[Dorigo 2005] Marco Dorigo and Christian Blum. *Ant colony optimization theory: A survey*. Theoretical computer science, vol. 344, no. 2, pages 243–278, 2005. (Cited on pages 3 and 23.)

[Draa 2011] Amer Draa, Souham Meshoul, Hichem Talbi and Mohamed Batouche. *A quantum-inspired differential evolution algorithm for solving the N-queens problem*. neural networks, vol. 1, page 12, 2011. (Cited on page 26.)

[Eiben 2003] Agoston E Eiben and James E Smith. Introduction to evolutionary computing. Springer Science & Business Media, 2003. (Cited on page 3.)

[Estévez 2009] Pablo Estévez, Michel Tesmer, Claudio Perez, Jacek M Zurada*et al. Normalized mutual information feature selection*. Neural Networks, IEEE Transactions on, vol. 20, no. 2, pages 189–201, 2009. (Cited on page 22.)

[Falcon 2011] Rafael Falcon, Marcio Almeida and Amiya Nayak. *Fault identification with binary adaptive fireflies in parallel and distributed systems*. In Evolutionary Computation (CEC), 2011 IEEE Congress on, pages 1359–1366. IEEE, 2011. (Cited on page 4.)

[Fayyad 1996] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth and Ramasamy Uthurusamy. *Advances in knowledge discovery and data mining*. 1996. (Cited on page 1.)

[Feynman 1982] Richard P Feynman. *Simulating physics with computers*. International journal of theoretical physics, vol. 21, pages 467–488, 1982. (Cited on pages 25 and 32.)

[Fister Jr 2012] Iztok Fister Jr, Xin-She Yang, Iztok Fister and Janez Brest. *Memetic firefly algorithm for combinatorial optimization*. arXiv preprint arXiv:1204.5165, 2012. (Cited on page 4.)

[Grover 1996] Lov K Grover. *A fast quantum mechanical algorithm for database search*. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 212–219. ACM, 1996. (Cited on page 4.)

[Hacine-Gharbi 2012] Abdenour Hacine-Gharbi, Philippe Ravier, Rachid Harba and Tayeb Mohamadi. *Low bias histogram-based estimation of mutual information for feature selection*. Pattern recognition letters, vol. 33, no. 10, pages 1302–1308, 2012. (Cited on page 21.)

[Hamming 1950] Richard W Hamming. *Error detecting and error correcting codes*. Bell System technical journal, vol. 29, no. 2, pages 147–160, 1950. (Cited on pages 49 and 70.)

[Han 2000] Kuk-Hyun Han and Jong-Hwan Kim. *Genetic quantum algorithm and its application to combinatorial optimization problem*. In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 2, pages 1354–1360. IEEE, 2000. (Cited on pages 5 and 26.)

[Han 2002] Kuk-Hyun Han and Jong-Hwan Kim. *Quantum-inspired evolutionary algorithm for a class of combinatorial optimization*. Evolutionary Computation, IEEE Transactions on, vol. 6, no. 6, pages 580–593, 2002. (Cited on pages 5, 26, 33, 34, 39, 41 and 54.)

[Han 2004] Kuk-Hyun Han and Jong-Hwan Kim. *Quantum-inspired evolutionary algorithms with a new termination criterion, H ε gate, and two-phase scheme*. Evolutionary Computation, IEEE Transactions on, vol. 8, no. 2, pages 156–169, 2004. (Cited on page 26.)

[Hanafi 1998] Saïd Hanafi and Arnaud Freville. *An efficient tabu search approach for the 0–1 multidimensional knapsack problem*. European Journal of Operational Research, vol. 106, no. 2, pages 659–675, 1998. (Cited on page 45.)

[Hassanzadeh 2011] Tahereh Hassanzadeh, Hakimeh Vojodi and Amir Masoud Eftekhari Moghadam. *An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm*. In Natural Computation (ICNC), 2011 Seventh International Conference on, volume 3, pages 1817–1821. IEEE, 2011. (Cited on page 4.)

[He 2006] Yuguo He. *An efficient attribute reduction algorithm*. In Intelligent Data Engineering and Automated Learning–IDEAL 2006, pages 859–868. Springer, 2006. (Cited on page 22.)

[Hipp 2000] Jochen Hipp, Ulrich Güntzer and Gholamreza Nakhaeizadeh. *Algorithms for association rule mining—a general survey and comparison*. ACM sigkdd explorations newsletter, vol. 2, no. 1, pages 58–64, 2000. (Cited on page 2.)

[Hoffmann 2011] Matthias Hoffmann, Moritz Mühlenthaler, Sabine Helwig and Rolf Wanka. Discrete particle swarm optimization for tsp: theoretical results and experimental evaluations. Springer, 2011. (Cited on page 3.)

[Holland 1992] John H. Holland. *Genetic algorithms*. Scientific american, vol. 267, no. 1, pages 66–72, 1992. (Cited on page 23.)

[Horng 2012] Ming-Huwi Horng. *Vector quantization using the firefly algorithm for image compression*. Expert Systems with Applications, vol. 39, no. 1, pages 1078–1091, 2012. (Cited on pages 4 and 58.)

[Hota 2010] Ashish R Hota and Ankit Pat. *An adaptive quantum-inspired differential evolution algorithm for 0–1 knapsack problem*. In Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on, pages 703–708. IEEE, 2010. (Cited on pages 5, 26, 29, 34, 39, 41 and 54.)

[Hu 2003] Keyun Hu, Yuchang Lu and Chunyi Shi. *Feature ranking in rough sets*. Artificial Intelligence Communications, vol. 16, no. 1, pages 41–50, 2003. (Cited on page 22.)

[Jati 2011] GK Jati. *Evolutionary discrete firefly algorithm for travelling salesman problem*. In ICAIS LNAI, page 393–403. Springer, 2011. (Cited on page 4.)

[Jensen 2005] Richard Jensen and Qiang Shen. *Fuzzy-rough data reduction with ant colony optimization*. Fuzzy sets and systems, vol. 149, no. 1, pages 5–20, 2005. (Cited on page 23.)

[Jourdan 2009] Laetitia Jourdan, Matthieu Basseur and E-G Talbi. *Hybridizing exact methods and metaheuristics: A taxonomy*. European Journal of Operational Research, vol. 199, no. 3, pages 620–629, 2009. (Cited on page 3.)

[Karaboga 2014] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk and Nurhan Karaboga. *A comprehensive survey: artificial bee colony (ABC) algorithm and applications*. Artificial Intelligence Review, vol. 42, no. 1, pages 21–57, 2014. (Cited on page 4.)

[Ke 2008] Liangjun Ke, Zuren Feng and Zhigang Ren. *An efficient ant colony optimization approach to attribute reduction in rough set theory*. Pattern Recognition Letters, vol. 29, no. 9, pages 1351–1357, 2008. (Cited on page 23.)

[Ke 2010] Liangjun Ke, Zuren Feng, Zhigang Ren and Xiaoliang Wei. *An ant colony optimization approach for the multidimensional knapsack problem*. Journal of Heuristics, vol. 16, no. 1, pages 65–83, 2010. (Cited on page 45.)

[Kellerer 2004] Hans Kellerer, Ulrich Pferschy and David Pisinger. Introduction to np-completeness of knapsack problems. Springer, 2004. (Cited on page 44.)

[Kennedy 1995] James Kennedy and Russell Eberhart. *Particle swarm optimization*. In Proceedings of IEEE conference on neural network, 1995. (Cited on pages 3, 11, 23, 26, 44 and 67.)

[Kennedy 1997] James Kennedy and Russell C Eberhart. *A discrete binary version of the particle swarm algorithm*. In Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, volume 5, pages 4104–4108. IEEE, 1997. (Cited on page 3.)

[Krause 2013] Jonas Krause, Jelson Cordeiro, Rafael Stubs Parpinelli and Heitor Silverio Lopes. *A survey of swarm algorithms applied to discrete optimization problems*. Swarm Intelligence and Bio-inspired Computation: Theory and Applications. Elsevier Science & Technology Books, pages 169–191, 2013. (Cited on page 3.)

[Kuo 2011] RJ. Kuo, MJ. Wang and TW. Huang. *An application of particle swarm optimization algorithm to clustering analysis*. Soft Comput, vol. 15, no. 3, page 533–542, 2011. (Cited on page 3.)

[Kwak 2002] Nojun Kwak and Chong-Ho Choi. *Input feature selection for classification problems*. Neural Networks, IEEE Transactions on, vol. 13, no. 1, pages 143–159, 2002. (Cited on pages 21 and 22.)

[Langley 1994] Pat Langley*et al.* Selection of relevant features in machine learning. Defense Technical Information Center, 1994. (Cited on page 22.)

[Layeb 2013] Abdesslem Layeb. *A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems*. Journal of Computational and Applied Mathematics, vol. 253, pages 14–25, 2013. (Cited on pages 38, 39 and 41.)

[Li 2010] Ju Li, Xing Wang and Xiaowen Fan. *Improved binary discernibility matrix attribute reduction algorithm in customer relationship management*. Procedia Engineering, vol. 7, pages 473–476, 2010. (Cited on page 22.)

[Liang 2010] Yanbing Liang, Linlin Liu, Dayong Wang and Ruijuan Wu. *Optimizing particle swarm optimization to solve knapsack problem*. In Information Computing and Applications, pages 437–443. Springer, 2010. (Cited on pages 45 and 62.)

[Lin 2010] Huang Lin, Xi Maolong and Sun Jun. *An improved quantum-behaved particle swarm optimization with binary encoding*. In Intelligent System Design and Engineering Application (ISDEA), 2010 International Conference, volume 1, pages 243–249. IEEE, 2010. (Cited on page 26.)

[Liu 2006] Jing Liu, Jun Sun and Wenbo Xu. *Quantum-behaved particle swarm optimization with adaptive mutation operator*. In Advances in Natural Computation, pages 959–967. Springer, 2006. (Cited on page 26.)

[Manju 2014] A Manju and Madhav J Nigam. *Applications of quantum inspired computational intelligence: a survey*. Artificial Intelligence Review, vol. 42, no. 1, pages 79–156, 2014. (Cited on pages 11, 67 and 68.)

[Marill 1963] Thomas Marill and David M Green. *On the effectiveness of receptors in recognition systems*. Information Theory, IEEE Transactions on, vol. 9, no. 1, pages 11–17, 1963. (Cited on page 23.)

[Marinakis 2013] Yannis Marinakis and Magdalene Marinaki. *Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem*. Soft Computing, vol. 17, no. 7, pages 1159–1173, 2013. (Cited on page 3.)

[Martello 2011] Silvano Martello, M Minoux, C Ribeiro and Gilbert Laporte. Surveys in combinatorial optimization. Elsevier, 2011. (Cited on page 3.)

[Neapolitan 2004] Richard Neapolitan and Kumarss Naimipour. *Jones and Bartlett Publishers,"*. 2004. (Cited on page 3.)

[Neshat 2014] Mehdi Neshat, Ghodrat Sepidnam, Mehdi Sargolzaei and Adel Najaran Toosi. *Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications*. Artificial Intelligence Review, vol. 42, no. 4, pages 965–997, 2014. (Cited on page 3.)

[Palit 2011] Supravo Palit, Saptarshi Neil Sinha, Mostafiz Amin Molla, Atreyee Khanra and Malay Kule. *A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm*. In Computer and Communication Technology (ICCCT), 2011 2nd International Conference on, pages 428–432. IEEE, 2011. (Cited on page 4.)

[Pat 2011] Ankit Pat, Ashish Ranjan Hota and Avneet Singh. *Quantum-inspired differential evolution on bloch coordinates of qubits*. In Advances in Computing, Communication and Control, pages 18–24. Springer, 2011. (Cited on page 5.)

[Pawlak 1982] Zdzisław Pawlak. *Rough sets*. International Journal of Computer & Information Sciences, vol. 11, no. 5, pages 341–356, 1982. (Cited on pages 19, 22 and 23.)

[Pawlak 2007] Zdzisław Pawlak and Andrzej Skowron. *Rough sets: some extensions*. Information sciences, vol. 177, no. 1, pages 28–40, 2007. (Cited on pages 19 and 23.)

[Pendharkar 2006] Parag C Pendharkar and James A Rodger. *Information technology capital budgeting using a knapsack problem*. International Transactions in Operational Research, vol. 13, no. 4, pages 333–351, 2006. (Cited on page 44.)

[Peng 2005] Hanchuan Peng, Fuhui Long and Chris Ding. *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, no. 8, pages 1226–1238, 2005. (Cited on page 21.)

[Pisinger 2005] David Pisinger. *Where are the hard knapsack problems?* Computers & Operations Research, vol. 32, no. 9, pages 2271–2284, 2005. (Cited on page 27.)

[Sayadi 2010] M Sayadi, Reza Ramezanian and Nader Ghaffari-Nasab. *A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems*. International Journal of Industrial Engineering Computations, vol. 1, no. 1, pages 1–10, 2010. (Cited on page 4.)

[Sayadi 2013] Mohammad Kazem Sayadi, Ashkan Hafezalkotob and Seyed Gholamreza Jalali Naini. *Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation*. Journal of Manufacturing Systems, vol. 32, no. 1, pages 78–84, 2013. (Cited on page 4.)

[Shannon 2015] Claude E Shannon and Warren Weaver. The mathematical theory of communication. University of Illinois press, 2015. (Cited on page 18.)

[Shor 1994] Peter W Shor. *Algorithms for quantum computation: Discrete logarithms and factoring*. In Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, pages 124–134. IEEE, 1994. (Cited on page 4.)

[Slowinski 2000] Roman Slowinski and Daniel Vanderpooten. *A generalized definition of rough approximations based on similarity*. IEEE Transactions on Knowledge & Data Engineering, no. 2, pages 331–336, 2000. (Cited on page 23.)

[Storn 1997] Rainer Storn and Kenneth Price. *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces*. Journal of global optimization, vol. 11, no. 4, pages 341–359, 1997. (Cited on pages 12, 13 and 26.)

[Su 2008] Haijun Su and Yupu Yang. *Quantum-inspired differential evolution for binary optimization*. In Natural Computation, 2008. ICNC'08. Fourth International Conference on, volume 1, pages 341–346. IEEE, 2008. (Cited on page 26.)

[Swiniarski 2003] Roman W Swiniarski and Andrzej Skowron. *Rough set methods in feature selection and recognition*. Pattern recognition letters, vol. 24, no. 6, pages 833–849, 2003. (Cited on page 23.)

[Talbi 2004] Hichem Talbi, Amer Draa and Mohamed Batouche. *A new quantum-inspired genetic algorithm for solving the travelling salesman problem*. In Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on, volume 3, pages 1192–1197. IEEE, 2004. (Cited on page 26.)

[Tavana 2013] Madjid Tavana, Kaveh Khalili-Damghani and Amir-Reza Abtahi. *A fuzzy multidimensional multiple-choice knapsack model for project portfolio selection using an evolutionary algorithm*. Annals of Operations Research, vol. 206, no. 1, pages 449–483, 2013. (Cited on page 44.)

[Tazuke 2013] Koichiro Tazuke, Noriyuki Muramoto, Nobuyuki Matsui and Teijiro Isokawa. *An application of quantum-inspired particle swarm optimization*

to function optimization problems. In Neural Networks (IJCNN), The 2013 International Joint Conference on, pages 1–6. IEEE, 2013. (Cited on page 5.)

[UCI 1998] *UCI Repository of machine learning databases*. http://ftp.ics.uci.edu/pub/machine-learning-databases, 1998. (Cited on page 73.)

[Wang 2001] Jue Wang and Ju Wang. *Reduction algorithms based on discernibility matrix: the ordered attributes method*. Journal of computer science and technology, vol. 16, no. 6, pages 489–504, 2001. (Cited on page 22.)

[Wang 2004] Guo-Yin Wang, Jun Zhao, Jiu-Jiang An and Yu Wu. *Theoretical study on attribute reduction of rough set theory: comparison of algebra and information views*. In Cognitive Informatics, 2004. Proceedings of the Third IEEE International Conference on, pages 148–155. IEEE, 2004. (Cited on page 22.)

[Wang 2007a] Haijun Wang, Shaoliang Wei and Yimin Chen. *An improved attribute reduction algorithm based on rough set*. In null, pages 1007–1010. IEEE, 2007. (Cited on page 22.)

[Wang 2007b] Xiangyang Wang, Jie Yang, Xiaolong Teng, Weijun Xia and Richard Jensen. *Feature selection based on rough sets and particle swarm optimization*. Pattern Recognition Letters, vol. 28, no. 4, pages 459–471, 2007. (Cited on page 23.)

[Wang 2007c] Yan Wang, Xiao-Yue Feng, Yan-Xin Huang, Dong-Bing Pu, Wen-Gang Zhou, Yan-Chun Liang and Chun-Guang Zhou. *A novel quantum swarm evolutionary algorithm and its applications*. Neurocomputing, vol. 70, no. 4, pages 633–640, 2007. (Cited on pages 5, 26, 34, 39, 41, 54, 70 and 80.)

[Wang 2011] Ling Wang, Mingde Zhang, Qun Niu and Jun Yao. *A modified quantum-inspired particle swarm optimization algorithm*. In Artificial intelligence and computational intelligence, pages 412–419. Springer, 2011. (Cited on page 26.)

[Wygralak 1989] Maciej Wygralak. *Rough sets and fuzzy sets-some remarks on interrelations*. Fuzzy Sets and Systems, vol. 29, no. 2, pages 241–243, 1989. (Cited on page 22.)

[Xiao 2008] Jing Xiao, YuPing Yan, Ying Lin, Ling Yuan and Jun Zhang. *A quantum-inspired genetic algorithm for data clustering*. In E— volutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pages 1513–1519. IEEE, 2008. (Cited on page 26.)

[Xue 2014] Bing Xue. *Particle Swarm Optimisation for Feature Selection in Classification*. 2014. (Cited on pages ix, 17 and 18.)

[Yang 2009a] Xin-She Yang and Suash Deb. *Cuckoo search via Lévy flights*. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pages 210–214. IEEE, 2009. (Cited on pages 3 and 67.)

[Yang 2009b] XS. Yang. *Firefly algorithms for multimodal optimization*. In Stochastic algorithms: foundations and applications, pages 169–178. Springer, 2009. (Cited on pages 3, 4, 14 and 44.)

[Yang 2010a] Xin-She Yang. Firefly algorithm : Nature-inspired metaheuristic algorithms. Luniver press, 2010. (Cited on pages 3, 4, 14, 15 and 44.)

[Yang 2010b] XS. Yang. *Firefly algorithm, stochastic test functions and design optimisation*. International Journal of Bio-Inspired Computation, vol. 2, no. 2, pages 78–84, 2010. (Cited on page 4.)

[Yang 2010c] XS. Yang. *A new metaheuristic bat-inspired algorithm*. In Nature inspired cooperative strategies for optimization (NICSO 2010), pages 65–74. Springer, 2010. (Cited on page 3.)

[Yousif 2011] A. Yousif. *Scheduling jobs on grid computing using firefly algorithm*. Journal of Theoretical and Applied Information Technology, vol. 33, no. 2, pages 155–164, 2011. (Cited on page 4.)

[Yu 2004] L. Yu and H. Liu. *Efficient feature selection via analysis of relevance and redundancy*. The Journal of Machine Learning Research, vol. 5, pages 1205–1224, 2004. (Cited on page 22.)

[Yuan 1999] H. Yuan, S. Tseng, W. Gangshan and Z. Fuyan. *A two-phase feature selection method using both filter and wrapper*. In Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on, volume 2, pages 132–136. IEEE, 1999. (Cited on page 22.)

[Zhou 2006] WG. Zhou, CG. Zhou, GX. Liu, HY. Lv and YC. Liang. *An improved quantum-inspired evolutionary algorithm for clustering gene expression data*. In Computational Methods, pages 1351–1356. Springer, 2006. (Cited on page 26.)

[Zouache 2015] Djaafar. Zouache, Farid. Nouioua and Abdelouahab. Moussaoui. *Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems*. Soft Computing, pages 1–19, 2015. (Cited on page 67.)

Bio-inspired Algorithms for Data mining
Abstract

Knowledge extraction from data bases, also called data mining, denotes the process of discovering useful, new and understandable knowledge from large data bases. Among data mining tasks where the nature of information extraction process may be seen as an optimization process. Consequently, we have needed fundamental approach different conventional approaches of knowledge discovery. This approach is based on inspiration, ideas and insights from the natural life to solve the knowledge extraction problems. In this thesis, we propose two novel bio-inspired algorithms to solve discrete optimization problems.
The first algorithm is called Quantum inspired Differential Evolution with Particle Swarm Optimization 'QDEPSO' which combines differential evolution, particle swarm optimization method and quantum-inspired evolutionary algorithm . In the initialization phase, the QDEPSO uses the concepts of quantum computing as the superposition state of qubits as well as the quantum measurement to represent and generate the diversity of the initial solutions. The second phase is an alternation between the DE operations and the adaptation of update formula of the velocity and the position of PSO algorithm.
The second algorithm is called Quantum-inspired Firefly Algorithm with Particle Swarm Optimization 'QIFAPSO'. This proposed algorithm uses the basic concepts of quantum computing such as superposition states of Q-bit and quantum measure to ensure a better control of the solutions diversity. Finally, the 'QIFAPSO' combine two strategies that cooperate in exploring the search space: the first one is the move of less bright fireflies towards the brighter ones and the second strategy is the PSO movement in which a firefly moves by taking into account its best position as well as the best position of its neighborhood. Finally, we propose the used QIFAPSO based on rough set for feature selection in classification.
Keywords: Data mining; Differential evolution; Quantum computing; Firefly algorithm; Particle Swarm Optimization; Feature selection; Rough set; Discrete problem optimization; Knapsack problem

Fouille de données basée algorithmes bio-inspirés

Résumé

L'extraction de connaissances dans les bases de données, également appelé "*data mining*", désigne le processus de découverte des informations et des connaissances utiles, nouvelles et compréhensibles à partir d'une base de données de grande taille, d'un entrepôt de données ou d'autres bases. La majorité des problèmes d'extraction de connaissances peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Par conséquent, nous avons besoin d'une approche fondamentale différente des approches d'extraction exactes classiques. Cette approche est basée sur l'inspiration des idées et des intuitions à partir de la nature et de la vie (biologique, physique, etc.) pour résoudre les problèmes d'extraction de connaissances.

Notre contribution est faite en deux phases : La première phase consiste à concevoir des méta-heuristiques bio-inspirés pour résoudre des problèmes d'optimisation combinatoire d'une manière générale et la deuxième phase consiste à réaliser et d'appliquer ces méta-heuristiques proposées aux problèmes d'extraction de connaissances.

Dans la première phase, nous avons proposé deux algorithmes bio-inspirés, le premier algorithme appelé QDEPSO hybride entre le DE et le PSO. Le deuxième algorithme appelé QIFAPSO fait coopérer le firefly algorithm et le PSO. Les deux algorithmes utilisent les concepts de l'informatique quantique. Dans la deuxième phase, nous avons appliqué l'algorithme QIFAPSO pour résoudre le problème de la sélection d'attributs.

Une évaluation expérimentale approfondie sur les différents jeux de données disponibles dans la littérature montre que les algorithmes développés sont performants et concurrents en terme de qualité de solutions comparant avec d'autres algorithmes qu'ont été développés pour résoudre des problèmes d'optimisation combinatoire ou bien dans la résolution de problème de la sélection d'attributs.

Mots clés : Fouille de données ; Informatique quantique ; algorithme de luciole ; Optimisation par essaime de particules; Sélection d'attributs ; Problème d'optimisation discrète ; Problème du sac à dos

منجمة المعارف اعتمادا على خوارزميات مستحات من الطبيعة

ملخـــــــــــص

منجمة المعـــارف من خلال قـاعدة البيانات تعني اكتشاف معلومـات جديدة و مفيدة، مـن خلال قواعـد بيانات واسعة. هناك طرق استخراج المعارف يمكن محاكتها على أنها عملية معقدة للتحسين التوافقي، لذلك نحتاج الى طرق جديدة في عملية استخراج المعارف تختلف عن الطرق الكلاسيكية، هذه الطرق تعتمد على النظر المعمق و استنباط الرؤى من الحياة الطبيعية.

في هذه الأطروحة اقترحنا أولا خوارزميتين مستوحاتين من الطبيعة لحل مشاكل التحسين التوافقي بصفة عامة: الخـوارزمية الأولى المقترحة هجـينة بين خـوارزمية التطور التفـاضلية و خـوارزمية الجسيمـات الأمثل، أمـا الخوارزمية الثانية المقترحة فهي هجينة بين خوارزمية اليراعـــة و خوارزمية الجسيمات الأمثل. كلا الخوارزميتين تعتمد في حل مشاكل التحسين التوافقي على مبادئ الحوسبة الكمومية، بعد ذلك قمنا باستغلال الخـــوارزمية الثانية المقترحة في حل مشكل من مشاكل التنقيب على المعارف ألا وهو اختيار الخصائص لقاعدة بيانات واسعـة، وقد استعملنا في هذا الحل نظرية المجموعة التقريبية في تقييم ملاءمة الخصائص المختارة.

وخلصنا في الأخير إلى نتائج مشجعة سواء بالنسبة لمشاكل التحسين التوافقي أو مشـــكل اختيار الصفـات، مقارنة بالحلول الموجودة على الساحة العلمية.

كلمات المفتاح: منجمة المعارف، خوارزميات مستحات من الطبيعة، مشـــاكل التحسين التوافقي، مشكـل اختيار الخصائص.