Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Multi-agent architecture for Multi-objective optimization of Flexible Neural Tree

Marwa Ammar [a,*], Souhir Bouaziz [a], Adel M. Alimi [a], Ajith Abraham [b,c]

[a] REsearch Groups in Intelligent Machines (REGIM), University of Sfax, National School of Engineers (ENIS), BP 1173, Sfax 3038, Tunisia
[b] Machine Intelligence Research Labs, WA, USA
[c] IT4Innovations, VSB-Technical University of Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 18 November 2015
Received in revised form
2 June 2016
Accepted 7 June 2016
Communicated by Zidong Wang
Available online 14 June 2016

Keywords:
Flexible Neural Tree
Multi-agent architecture
Multi-objective optimization
Evolutionary Computation algorithms
Negotiation
Classification

ABSTRACT

In this paper, a multi-agent system is introduced to parallelize the Flexible Beta Basis Function Neural Network (FBBFNT)' training as a response to the time cost challenge. Different agents are formed; a Structure Agent is designed for the FBBFNT structure optimization and a variable set of Parameter Agents is used for the FBBFNT parameter optimization. The main objectives of the FBBFNT learning process were the accuracy and the structure complexity. With the proposed multi-agent system, the main purpose is to reach a good balance between these objectives. For that, a multi-objective context was adopted which based on Pareto dominance. The agents use two algorithms: the Pareto dominance Extended Genetic Programming (*PEGP*) and the Pareto Multi-Dimensional Particle Swarm Optimization (*PMD_PSO*) algorithms for the structure and parameter optimization, respectively. The proposed system is called Pareto Multi-Agent Flexible Neural Tree (*PMA_FNT*).

To assess the effectiveness of *PMA_FNT*, four benchmark real datasets of classification are tested. The results compared with some classifiers published in the literature.

1. Introduction

In Artificial Intelligence field, scientists have always meditated natural life and behaviors to invent and evolve intelligent systems. We can for instance mention the Artificial Neural Network (*ANN*) which was inspired from the human brain behavior, the Evolutionary Computation (*EC*) algorithms which were inspired from some natural phenomena and so on [1].

Thanks to its efficiency, the Multi-Layer ANN and in particular the well known ANN' topology has attracted more attention nowadays and it is now considered as a powerful system for complex search problems such as prediction [2], pattern recognition [3] and classification [4,5].

In fact, the main weakness of the Neural Network training is the slow convergence to the near desired output, specially with large problems. To circumvent this weakness, many works hover around the objective of accelerating the NN training with ameliorating its performance. Some researchers focus on only training the output weights for a very large neural network [6,7]. Others tried to find some alternatives to improve the MLPs performance. In this context, Evolutionary Computation EC has been considered as a good candidate for the ANN evolution [8]. EC includes Swarm Intelligence like Particle Swarm Optimization (*PSO*) [9], Evolutionary Algorithms such as Genetic Algorithm (*GA*) [10] as well as some Mimic Algorithms like Harmony Search (*HS*) [11] and so on.

In addition, it is not possible to predict an ideal ANN structure likely to resolve all the problems. So, a structure adaptive task is required to improve the Neural Network performance. Decreasing the complexity of the NN structure and increasing its flexibility has led to the establishment of a new ANN encoding based on the tree representation called Flexible Neural Tree (FNT) [3]. The FNT topology is adapted in our work using the Beta function as a transfer function (FBBFNT). Although the FBBFNTs could solve complex problems [12], this model suffers from high time-cost. In addition, real problems include a large set of features/inputs which increase the time complexity. So, we have considered parallelizing the learning process of the FBBFNT so that we can deal with real problems with respect to time cost.

On the other side, Multi-Agent System (*MAS*) is viewed as a new intelligent system inspired from the social system. In fact, as an intelligent system, the human being is a part of a social system in which he operates and interacts with other humans distributed in the same environment. Much in the same way, MAS distributes and coordinates a set of jobs, tasks and decisions between different entities, called agents, to build coherent and interactive systems [13].

* Corresponding author.
E-mail addresses: marwa.ammar.tn@ieee.org (M. Ammar), souhir.bouaziz@ieee.org (S. Bouaziz), adel.alimi@ieee.org (A.M. Alimi), ajith.abraham@ieee.org (A. Abraham).

http://dx.doi.org/10.1016/j.neucom.2016.06.019
0925-2312/© 2016 Elsevier B.V. All rights reserved.

In this work, the Multi-Agent System is used to optimize and parallelize the learning process of the evolving Flexible Beta Basis Function Neural Tree (FBBFNT) model. It uses a set of interacting agents; a Structure Agent for NN architecture/structure optimization and a set of Parameter Agents for NN parameters optimization.

Indeed, an agent is an independent and autonomous entity that has the ability to interact, cooperate, coordinate and negotiate with each other. For that, a communication process is disposed to ensure a good negotiation between agents. In our model, a negotiation protocol is implemented as a communication protocol. It ensures the exchange of information between agents which compete to find the optimal ANN for the treated problem. The negotiation strategy is based on two factors; the Agent Dominance Rate (*ADR*) and the Agent Trust Value (*ATV*). They ensure the evaluation of the agent performance in the multi-agent system.

To attain the optimal solution, this model takes into consideration two main objectives: the accuracy and the structure complexity of the neural network. In fact, the trade-off between these objectives is caused by their influence on the convergence and the effectiveness of the given solution. This conflict is implicitly pointed out in [14,2]. A simple aggregation of the two objective functions has averted systems dealing with the conflict. However, these objectives have different impact on the NN and should ensure good balance between them. Consequently, we developed a multi-objective optimization to solve this trade-off. Indeed, the learning process adopted a multi-objective optimization based on the Pareto dominance. Learning agents use multi-objective evolutionary algorithms; the Pareto dominance Extended Genetic Programming algorithm (PEGP) and a Pareto Multi-Dimensional Particle Swarm Optimization algorithm (PMD_PSO) to optimize the FBBFNT structure and parameters respectively. According to the formerly mentioned, the model was called the Pareto Multi-Agent Flexible Neural Tree PMA_FNT. It will be described in Section 3. The functionalities and the training method of Structure and Parameter agents are detailed in Section 3.1 and 3.2, respectively. The communication process including the strategy and the negotiation protocol used is described in Section 3.3. In Section 4, the experimentation results with real classification problems and a comparative study with other classifiers are presented. The final section introduces some concluding remarks.

2. Related works of ANN' parallel training

Among the first attempts in parallelizing neural network training, the model, which is presented in [15,16], distributes neurons across cluster nodes working in parallel. This method relies on expensive and complex hardware. From the Network Parallel Training to the Pattern Parallel Training, Suri et. al. [17] and Dahl et. al [18] used a duplicated ANN at each node to train a subset of patterns from the training set in parallel. In [19], Quteishat et al. presented an ANN based on multi-agent classifier system. These agents work in parallel and form two interacted agent teams where the communication strategy followed the TNC (trust, negotiation, communication) reasoning model. Learning agents were trying to adjust the weights vector of the NN to provide an accurate classifier for the given patterns. Thus, optimizing the connection weights of the network could largely improve its performance. As a result, many researchers expressed interest in integrating the evolving population based algorithms as a framework for learning tasks [20,8]. In this work, a set of agents is dedicated for parameters optimization using a swarm intelligent algorithm. These agents could manipulate different neural network structures in the same swarm assuming that it is wiser to preserve a variety of neural networks rather than just the 'best' one. Moreover, the best NN structure could not be predicted before exploring the search space. This hard task was authorized to another agent, called Structure Agent, in our multi-agent model. Hence, our model focuses on parallelizing the learning process of a multi-hidden layer NN including structure evolving and parameters evolving. It uses a set of interacted agent for different tasks forming a multi-agent system. The model adopted a communication process between agents organized by a negotiation protocol to lead the system to the optimal ANN with less time and cost.

3. Pareto multi-agent flexible neural tree PMA_FNT

At the beginning, the fundamental characteristics of the used neural network might be described. Our model belongs to the Multi-Layer Preceptor (*MLP*) ANNs. It has two hidden layers which contain a set of functional nodes that use the Beta function [21] as a transfer function, as well as a set of terminal nodes. While the output layer contains one linear functional node, the input layer is composed of a set of terminal nodes. It also adopts the tree encoding for a flexible handling with the ANN optimization. It is called the Flexible Beta Basis Function Neural Tree (FBBFNT). The FBBFNT, as a universal approximator [22], has proved its efficiency with benchmark prediction problems. Starting with a random population of feasible ANNs, the FBBFNT model goes through a learning/optimization phase in order to reach the optimal ANN. The ANN optimization consists of two processes: the ANN structure evolution and the ANN parameter evolution (see Fig. 1). In general, to seek the optimal ANN for a given problem is a hard task that requires a long training time. The iterative ANN optimization process reinforces the dependency between learning tasks and increases time cost. This computation challenge could be overcome by introducing parallel algorithms/systems [18]. The *PMA_FNT* distributes and parallelizes the optimization task between the Structure Agent and the Parameter Agents (see Fig. 2). They work in cooperation and communicate through the negotiation protocol. For the optimal solution to be accurate, it should have the optimal structure with the optimal parameters. Indeed, the optimal structure can be defined as the best distribution of nodes by layers with fewer number of functional nodes. As previously mentioned, this model aims at looking for the best FBBFNT architecture with the best accuracy rate. In fact the evaluation of the solution should meet the two systems' objectives. As far as the structure complexity is concerned, an adequate function is used to report the structure complexity of the network. Previously, the number of node was the principal measure for the network structure [3,14,2]. In our work, we are suggesting a new function $f1$ that measures the congestion rate of terminal and functional nodes by layers. In general, it computes the External Mean Depth (*EMD*) of the Tree (*T*) multiplied by the number of functional nodes (*NFN*) (see Eq. (1)).

$$f1(T) = EMD(T) * NFN(T)$$
$$EMD(T) = EPL(T)/NL$$
$$EPL(T) = \sum_{i=1}^{NL} path(node_i) \tag{1}$$

where NL is the number of leaves/terminal nodes. In addition, the accuracy of a model is an important factor to measure its performance.

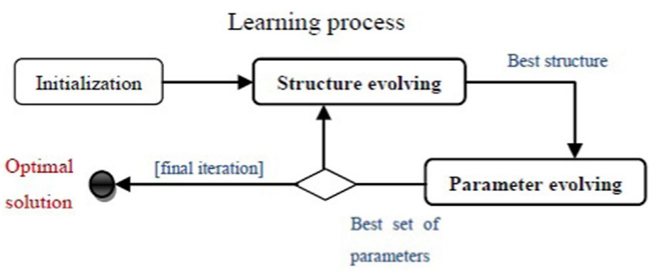


Fig. 1. FBBFNT evolving process.

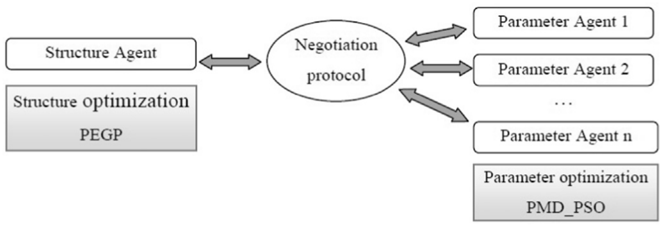


Fig. 2. The global architecture of *PMA_FNT*.

The function $f2$ computes the classification rate of the FBBFNT (see Eq. (2)).

$$f2(FBBFNT) = \frac{1}{P}\sum_{j=1}^{P}(Y_{out}^{j} = Y^{j}) \tag{2}$$

where P value represents the number of samples.

3.1. Pareto dominance extended genetic programming for structure agent

In this work, the optimal structure or the almost-optimal structure is achieved by using the Pareto dominance Extended Genetic Programming (*PEGP*) algorithm. It is an extended version of The Genetic Programming (*GP*) paradigm introduced by Koza [23]. The PEGP algorithm involves three operators: selection, crossover and mutation.

- *Selection operator*: It is used to select two individuals from the population Pt as a parent of the new child procreated by crossover/mutation operator.
- *Crossover operator*: It is the swapping operation of two sub-trees from two different individuals randomly selected.
- *Mutation*: Four different mutation operators are used in the PEGP to generate offspring from the parents. These mutation operators work as follows: changing one terminal node; changing all the terminal nodes; growing (replacing randomly a leaf node in hidden layer by a sub-tree); pruning (replacing randomly a beta operator node by a leaf node).

As the genetic operators rely on a probabilistic aspect to choose the candidate parents and to apply each modification on an arbitrary part of the FBBFNT, there is no guarantee that the offspring FBBFNT performs better than its parents. So, the outcome needs an evaluation before joining the population. Respecting the objectives of the *PMA_FNT* model, each solution is controlled by the two functions $f1$ and $f2$. Facing the trade-off between architecture complexity and accuracy, the offspring solution can join the population when it satisfies this dominance condition; for each FBBFNT candidate, if it dominates at least two solutions in the current population, then it will take the place of the worst solution in this population. So, the Structure Agent executes the Pareto dominance Extended Genetic Programming (*PEGP*) algorithm as follows:

Algorithm 1. Pareto dominance Extended Genetic Programming PEGP.

```
1:  $Evaluate(population) \leftarrow [f1(population), f2(population)]$
2:  repeat
3:    if $operator$ is $crossover$ then
4:      $(parent_1, parent_2) = SelectParents(population, size_{population})$
5:      $(child_1, child_2) = crossover(parent_1, parent_2)$
6:      $Evaluate(child_1) \leftarrow [f1(child_1), f2(child_1)]$
7:      if $DOMINANCE(child_1, population)$ is $true$ then
8:        Remove the worst individual from population
9:        $Add(child_1, population)$
10:     end if
11:     $Evaluate(child_2) \leftarrow [f1(child_2), f2(child_2)]$
12:     if $DOMINANCE(child_2, population)$ is $true$ then
13:       Remove worst individual from population
14:       $Add(child_2, population)$
15:     end if
16:   end if
17:   if $operator$ is $mutation$ then
18:     $(parent) = SelectParents(population, size_{population})$
19:     $(child) = crossover(parent)$
20:     $Evaluate(child) \leftarrow [f1(child), f2(child)]$
21:     if $DOMINANCE(child, population)$ is $true$ then
22:       Remove worst individual from population
23:       $Add(child, population)$
24:     end if
25:   end if
26: until termination condition is satisfied
27: $Fronts_{set} = FastNonDominationSort(population)$
28: return $Fronts_{set}$
```

Algorithm 2. DOMINANCE.

```
    function DOMINANCE $Indiv_{new}, population = (p_1, p_2, \dots p_{size_{population}})$
2:    $state_{domin} \leftarrow false$
      $dominatedSolutions \leftarrow 0$
4:    $i \leftarrow 1$
      repeat
6:      if $Indiv_{new}$ domine $p_i$ then
          $(dominatedSolutions = dominatedSolutions + 1)$
8:      end if
        $i = i + 1$
10:   until $(dominatedSolutions = 2)$ or $(i = size_{population})$
      if $dominatedSsolutions = 2$ then
12:   $state_{domin} \leftarrow true$
      end if
14:   return $state_{domin}$
    end function
```

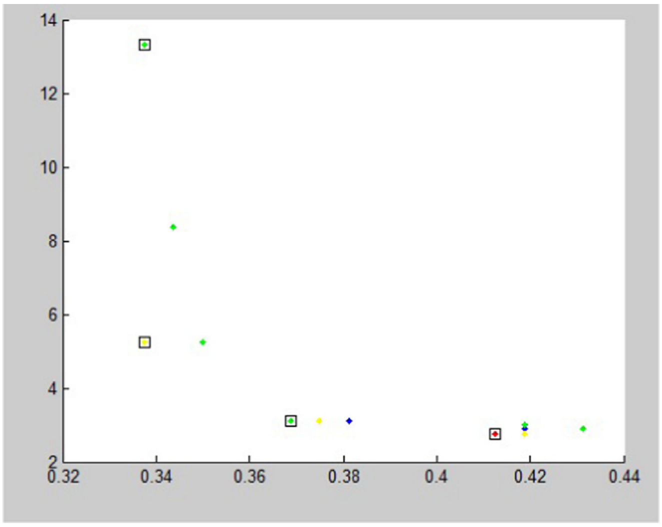


Fig. 3. Evolution of solutions in different fronts after a parameter optimization *This figure shows an internal state if the ith iteration when four parameter agents sent their best solution to the structure agent for bi-objective evaluation (accuracy, architecture complexity).*

At the end of the structure optimization, an evolved population is obtained and formed by a number of fronts (where $front_1$ is the Pareto front) using the FastNonDominationSort algorithm. This algorithm is developed by the well known multi-objective algorithm $NSGA-II$ [24]. The number of fronts corresponds to the number of Parameter Agents in the system. In fact, each agent treats a set of solutions that belong to one front. Our model does not focus on one solution in the population to optimize its parameters, but, it aims at giving the same opportunity to all solutions in the population in order to enlarge the search space and discover better feasible solutions. Experimentally, this method has proved to be efficient. Some solutions presented in $front_{i+k}$ can dominate others in $front_i$ after a parameter optimization phase. Fig. 3 shows a study case of four fronts presented in different colors that correspond to the set of Pareto solutions given by four Parameter agents. At this level, after collecting the agents' outcomes, the model selects the solutions that formed the Pareto front of the global system (black squares). So, the fact that the resulted Pareto front is formed by different colors means that solutions from different agents are competing to be the optimal one.

3.2. Pareto multi-dimensional particle swarm optimization for parameter agent

After the structure evolution phase, the FBBFNT model needs an adjustment of its set of parameters to improve its performance. The set of ANN parameters is concerned with the Beta function parameters (center: c, spread: s and the form parameters: p and q) and the connecting weights of the ANN. In this phase we have chosen to apply the powerful evolutionary algorithm which is the Particle Swarm Optimization (PSO) algorithm. The shortcoming of the latter is that it only works only in fixed dimension search space. However, the parameter agent in this model manages to work with different network configurations. It is supposed to optimize the parameters of the set of FBBFNTs belonging to a designed front which form a heterogeneous population in term of dimension. Fortunately, in 2009, Kiranyaz et al. [25] introduced a new version of PSO likely to optimize the position of particles with different dimensions. It was called the multi-dimensional PSO ($MDPSO$). Initially, the FBBFNT is decoded into a matrix containing the set of parameters of the transfer function in the functional nodes with the set of connecting weights. Then, these matrices are duplicated into a set of similar matrices containing random values. These elements form the howl population. In fact, elements with the same index present a particle in the Swarm and they are considered as the different possible dimensions of this particle (see Fig. 4). As it is known in the basic concept of the PSO algorithm, the particle converges to the global optimum using its best personal position and the position of the best global particle. In our case, when the search space is multi-dimensional, the particle a needs to keep track of its best personal dimension $\tilde{xd_a}$ that corresponds to the best memorized personal position $xy_a^{\tilde{xd}_a(t)}$. In addition, the particle memorizes its best position $xy_a^{xd_a(t)}$ in a particular dimension (see Fig. 5(left)). This information is useful when the particle navigates again with the same dimension. Moreover, the swarm needs to memorize some information about the global best positions xy_{gbest}^{dbest} in the best global dimension. Furthermore, it is supposed to know the best position in each dimension (see Fig. 5(right)). The objective is to seek out the optimum positions in all dimensions while navigating in a multi-dimensional search space. In each iteration, particles try to fly with one of its possible dimensions to reach the optimum. Then, the parameter agent evaluates the particles' performance through two levels: First, it evaluates the new positions to update the local best for each particle and the global best of the swarm. Second, it evaluates the dimension used by each particle to determine the best dimension representing this particle and the global best dimension that corresponds to the global best particle in the swarm (see Algorithm 3). After a maximum number of iterations, the resulted swarm contains particles with the best positions in the different dimensions. In this phase, a Pareto dominance algorithm is applied on the swarm based on the fitness of particles ($f2$ function) and the dimension of particles ($f1$ function). It looks for the best non-dominated solution in the search space. This set of solutions is then ready to be sent as a response. The applied algorithm is called the Pareto Multi-Dimensional Particle Swarm Optimization PMDPSO (see Algorithm 3).

Algorithm 3. Pareto Multi-Dimensional Particle Swarm Optimization.

```
    for ∀ $a \in S$ do
       $sd_a(0) \leftarrow dimension(a)$
3:     $xx_a^{xd_a(0)}(0) \leftarrow parameters(a)$
       $vd_a(0) \leftarrow rand,\ vx_a^{xd_a(0)} \leftarrow rand$
       $\tilde{xd}_a(0) = xd_a(0),\ xy_a^{xd_a(0)}(0) = xx_a^{xd_a(0)}(0)$
6: end for
       ▷** the function $f = 1 - f2$ supposed to be
     the fitness function for the particle evaluation**
    for ∀ $t \in [1, IterMax]$ do
      for ∀ $a \in S$ do
          ▷** update of the local best position reached
     by the particle $a$ in its current dimension $xd_a(t)$**
9:        if $f(xx_a^{xd_a(t)}(t) < f(xy_a^{xd_a(t)}(t-1))$ then
            $xy_a^{xd_a(t)}(t) = xx_a^{xd_a(t)}(t)$
          ▷** update of the best particle of the global
     best position for the dimension $xd_a$ (t)**
          if $f(xx_a^{xd_a(t)}(t) < f(xy_{gbest(xd_a(t))}^{xd_a(t)}(t-1))$ then
12:           $gbest(xd_a(t)) = a$
            $x\tilde{y}^{xd_a(t)}(t) = xy_{gbest(xd_a(t))}^{xd_a(t)}(t) = xx_a^{xd_a(t)}(t)$
              ▷ ** update of the local best dimension
     used by the particle $a$**
            if $f(xx_a^{xd_a(t)}(t) < xy_a^{\tilde{xd}_a(t-1)}(t-1)$ then
15:             $\tilde{xd}_a(t) = xd_a(t)$
                ▷** update of the global best dimension
     in the swarm**
            if $f(xx_a^{xd_a(t)}(t) < f(x\tilde{y}_{dbest}(t-1)))$ then
              $dbest = xd_a(t)$
18:           end if
          end if
        end if
21:   end if
   end for
    if the termination criteria are met then
24:   Stop
      else
       $\psi(t) = ((MaxIter - t) * (\psi_{start} - \psi_{end})/(MaxIter - 1) + \psi_{end}$
27:      for ∀ $a \in S$ do   ▷** Compute the new position
     of each particle in the swarm according to the local
     and global best position for the $xd_a(t)$ dimension**
             $vx_a^{xd_a(t)}(t+1) = \psi(t)vx_a^{xd_a(t)}(t) + c1.\ r1(t)(xy_a^{xd_a(t)}(t)$
             $-xx_a^{xd_a(t)}(t)) + c2.\ r2(t)(x\tilde{y}^{xd_a(t)}(t) - xx_a^{xd_a(t)}(t))$
             $xx_a^{xd_a(t)}(t+1) = xx_a^{xd_a(t)}(t) + vx_a^{xd_a(t)}(t+1)$
       ▷** Compute the new dimension of each particle in the
      swarm according to the local best dimension of the
     particle and the global best dimension in the swarm**
```

```
30:  $vd_a(t+1) = \lfloor vd_a(t) + c1.\, r1(t)(x\tilde{d}_a(t) - xd_a(t)) + c2.\, r2(t)(dbest - xd_a(t)) \rfloor$
     $xd_a(t+1) = xd_a(t) + vd_a(t+1)$
       end for
33:      end if
   end for
   $Fronts_{set} = FastNonDominationSort(population)$
36: return $ParetoFront = Fronts_{set}(1)$
```

3.3. Communication process

The communication process is a collection of techniques that establish coherent interactions between agents in the system. Generally speaking, negotiation is an interaction mechanism that aims at resolving a conflict of interest between two or more parts (agents) to reach mutually beneficial deals [26]. It is a means of communication used according to the defined protocol and strategies of the agents. According to the context of use, the automated negotiation belongs to one of these categories: game theory, heuristic and argumentation based. In negotiation, there are three variables to consider: number of negotiators (one-to-one, one-to-many or many-to-many), the number of issues (single or multiple) and the manner how to deal with these issues (issue-by-issue or package-deal) [27]. In this work, the communication is based on a number of bilateral one-to-one negotiations between the Structure Agent (as an initiator) and a Parameter Agent (as a participant). In addition, as the agent adopts a multi-objectives optimization, the *PMA_FNT* deals with multi-issues negotiations.

3.3.1. Strategy

The strategy consists of mapping the agent state and the received information onto the next action that is taken during negotiation. The decision of the agent for the next step in this model is mainly based on the Agent Dominance Rate (*ADR*). This rate evaluates the agent performance and contributes to determining the '*winner*' and the '*loser*' agent. The strategy adopted by the initiator agent is formulated in the following set of rules:

- Reject-proposal If [(the agent is not the winner) and (the stopping criterion was reached)].
- Quit If [(the number of fronts given by the SA was reduced) and (the agent is the loser)].
- Accept-proposal If [(the agent is the winner) and (the stopping criterion was reached)].
- Counter-proposal If (the stopping criterion was not reached yet).

In this model, the ADR is defined as the rate of the number of nondominated solutions given by the agent per the total number of solutions (see Eq. (3)). It measures the temporary agent performance in each negotiation round. So, the system needs another factor to illustrate the performance of each agent throughout the negotiation session. It is the Agent Trust Value (ATV). The ATV factor represents a control parameter of the degree of trust offered by the system (see Eq. (4)). A simple comparison between ATVs allows the system to determine the winner and the loser agent. The winner agent is awarded by the set of solutions of the Pareto front for parameter optimization. Therefore, the system trusts the 'winner' to provide the optimal final solution (Accept-proposal) in the final round/ global iteration.

$$ADR_i = \frac{nondominated - solutions - number(PA_i)}{total - solutions - number} \tag{3}$$

$$ABV_i(t+1) = \alpha.ABV_i(t) + \beta.ADR_i \tag{4}$$

where $i \in [1..N]$// N is the number of PAs On the other hand, our model ensures a multi-objective structure optimization where the number of resulted fronts is variable. Thus, if the front numbers increase, new agent(s) will be added to the global system. Otherwise (fronts number decreased), the system looks for the loser agent and destroys it (Quit) until we obtain the needed agents for the next round.

3.3.2. Negotiation protocol

The exchange of messages between agents is governed by a communication protocol. The adopted protocol for the *PMA_FNT* is a reviewed version of the Iterated Contract Net Interaction Protocol (see Fig. 6). It is the iterated version of the Contract Net Protocol defined by Smith and Davis [28]. The negotiation process includes the different interactions between agents according to the used protocol. When facing a conflict, the agent executes various possible alternatives or scenarios.

In our system, the negotiation session starts when the Structure Agent as an initiator sends a message to all Parameter Agents as participants, requesting a parameter optimization for a set of solutions in the corresponding front ('*C-PO*'). The participant informs the initiator of its agreement and starts the parameter optimization process during a fixed period ($T < deadline$). In parallel, when the initiator receives the approval for its requests, it applies a structure optimization for the current population of ANN. After that, all participants send their proposals.

At this time, SA evaluates the agent performance using ADR and ATV. This evaluation is based on the comparison between the set of solutions provided by the PAi when optimizing the frontj and the set of solutions provided by SA of the $front_j$. (For example, if the PA_i has optimized the solutions in the front 2 then, their result solutions will be compared with solutions of the front 2 given by the SA in the next round.) Thus, we disposed a Pareto dominance comparison between the two sets according to the architectural complexity and the accuracy presented by the error made. Besides, we looked for the number of nondominated solutions from the PA_i to compute the *ADR* value. Only the nondominated solutions will join the population as the new front replacing the previous one.

After the ATV update, the initiator checks the stopping criteria corresponding to the union of two conditions. The first is {if the error made by the optimal solution found so far is smaller than a fixed threshold}. The second is related to the {maximum number of global iterations}.

- When the stopping criterion is valid (i.e. minimum one condition was reached), the system is in the final iteration to provide its final evolved FBBFNT solving the treated problem. In this case, the initiator responds to the winner agent by '*accept − proposal*'. This message informs the agent that the system considers its best solution as the optimal final solution. The other agents receive the '*reject − proposal*' message and the negotiation session is then closed.
- In the opposite case (not final iteration), three alternatives are possible:
  - If the number of the resulted fronts from the last structure optimization is the same, the initiator responds to all participants by a '*counter-proposal*'. The counter-proposal message contains a novel set of solutions taken from the front in the current population. This message gets back the scenario to the previous step. It triggers another optimization phase and another negotiation round.
  - If the number of fronts is reduced, the system will reject the loser agent (*Quit*) and search for the needed agents for the next round.
  - If the number of fronts increases, the system will recreate new agents for the extra fronts ('*C-PO*').

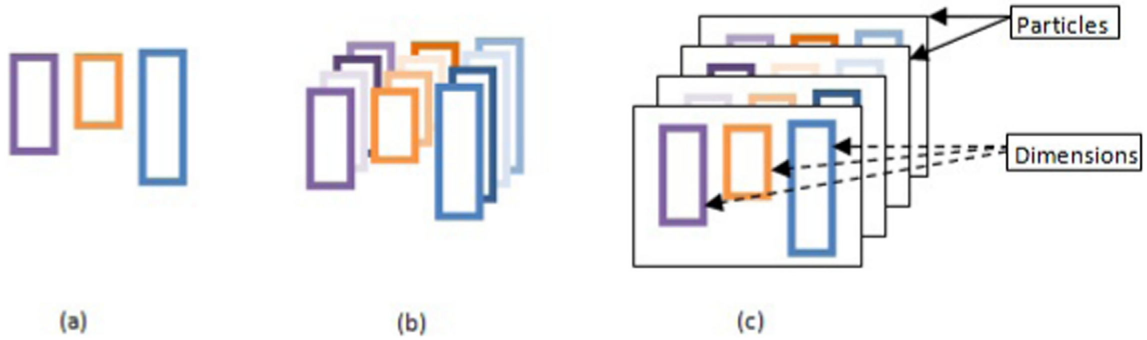


Fig. 4. Example of a population generation for the MD PSO algorithm: (a) representation of FBBFNTs with different structures (from one front), (b) representation of the population of FBBFNTs for parameter optimization with different set of parameters, and (c) representation of different particles in the Swarm.

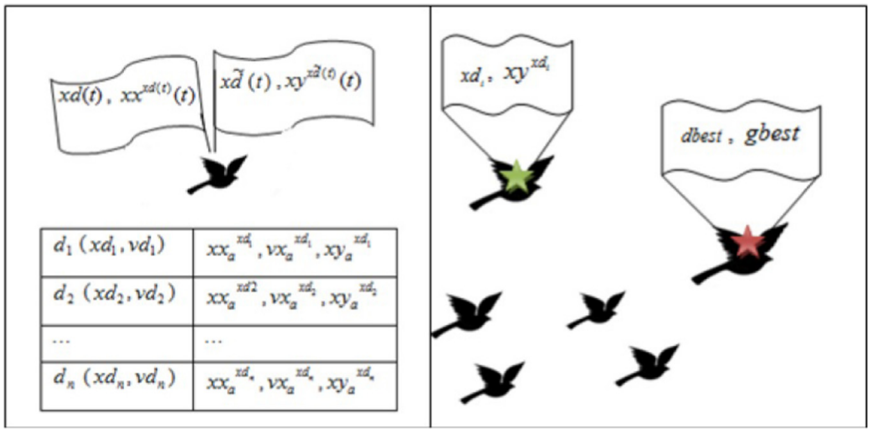

$xd(t), xx^{xd(t)}(t)$
$\tilde{xd}(t), xy^{\tilde{xd}(t)}(t)$
$d_1\ (xd_1, vd_1)$
$xx_a^{xd_1}, vx_a^{xd_1}, xy_a^{xd_1}$
$d_2\ (xd_2, vd_2)$
$xx_a^{xd_2}, vx_a^{xd_2}, xy_a^{xd_2}$
...
...
$d_n\ (xd_n, vd_n)$
$xx_a^{xd_n}, vx_a^{xd_n}, xy_a^{xd_n}$
xd_i, xy^{xd_i}
$dbest, gbest$

Fig. 5. The memory of the particle (left) and the swarm (right) in the multi-dimensional PSO.

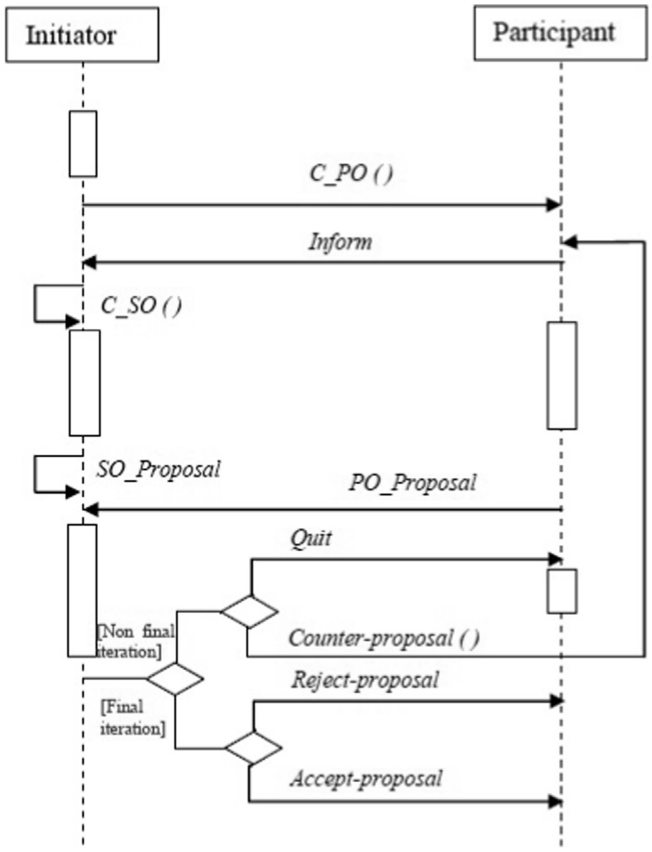


Fig. 6. Sequential diagram of the negotiation protocol.

Table 1
Four real datasets used for evaluating PMA_FNT performance.

Dataset	Samples	classes	Features	Selected deatures
Colon cancer	62	2	2000	26
Leukemia	72	2	7129	36
DLBC Lymphoma	47	2	4026	80
Lung cancer	181	2	12 533	40

Table 2
Parameters setting.

Algorithm	Parameter	Initial value
Pareto Extended Genetic Programming Algorithm (PEGP)	Population size	50
	Crossover probability	0.3
	Mutation probability	0.6
	Generation gap	0.9
Pareto Multi-dimensionel Particle Swarm Optimization (MD PSO)	Population size	50
	c1	0.2
	c2	0.2
	$\psi_s tart$	0.9
	$\psi_e nd$	0.4
Communication protocol	α	0.2
	β	0.8

4. Experimentation

Our system was implemented in Matlab platform using the distributed computing toolbox and the parallel computing toolbox to design the multi-agent system. In this section, the proposed

Table 3
The average results of the proposed model generated after 20 runs.

Dataset	Training accuracy (%)	Testing accuracy (%)	RMSE	Confusion matrix of worst test			
				TP	TN	FP	FN
Leukemia	99.71	99.68	0.123	19	13	1	1
Lymphoma	100	99.72	0.108	9	10	0	1
Colon Cancer	99.67	99.36	0.146	10	21	1	1
Lung Cancer	100	99.74	0.115	132	14	1	1

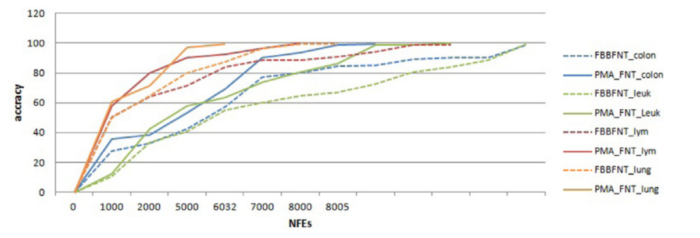


Fig. 8. Comparison of time complexity between FBBFNT and our model.

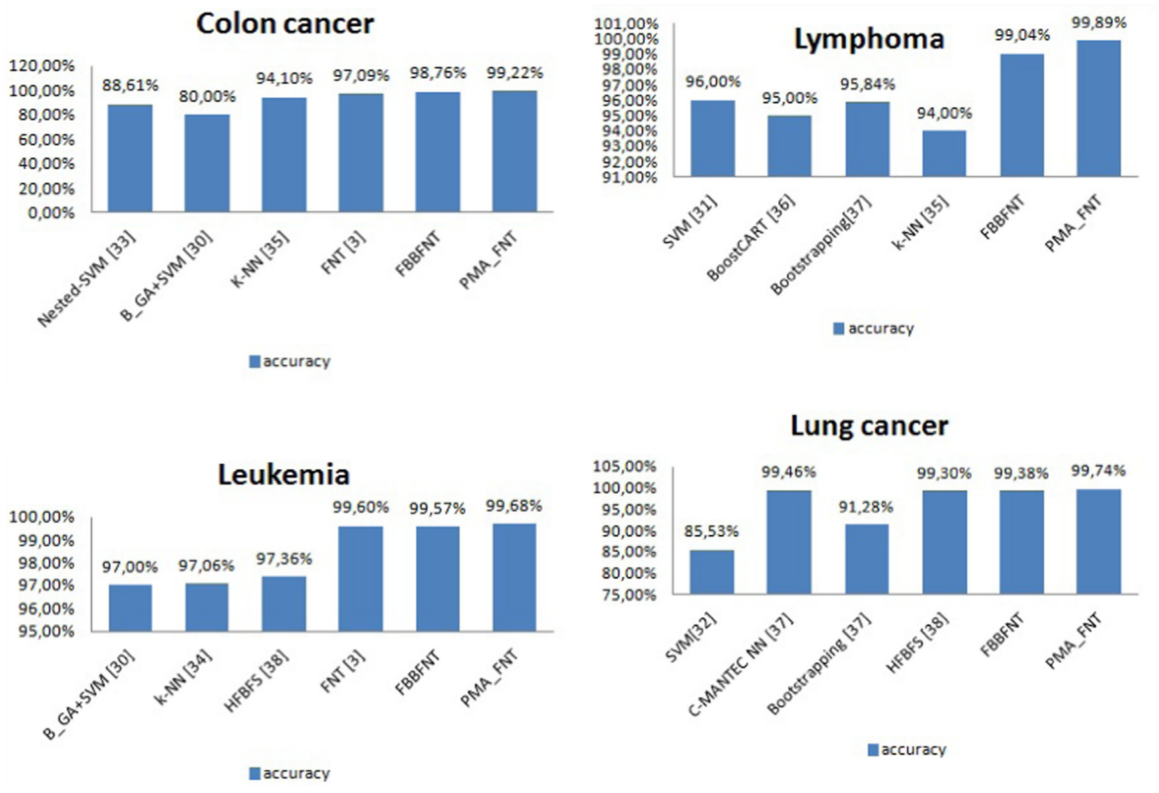


Fig. 7. Comparison of classification performance between PMA_FNT and other classifiers for four datasets.

model is applied to some real classification datasets to evaluate its performance. These well known datasets (such as Leukemia, Colon Cancer and Lymphoma) are high dimensional. For this reason, a feature selection phase was interfered to reduce the number of features and prevent our system from misleading information. Then, it is compared to some existing works from the literature.

4.1. Data source[1]

4.1.1. Colon cancer dataset

In cancer research, the Colon cancer ranks second in most mortal cancers in Western countries. It is a malignant tumor that is born in the cells of the colon. The colon cancer microarray dataset contains 62 samples of gene expression information extracted from 40 tumors and 22 normal colon tissues [29]. It was analyzed with microarray containing more than 6500 human genes. We decided to use 2000 genes (features) where 31 samples were considered as training set and the remaining 31 samples were considered as a testing set.

[1] Available at: http://sdmc.lit.org.sg/GEDatasets/Datasets.html.

4.1.2. Leukemia dataset

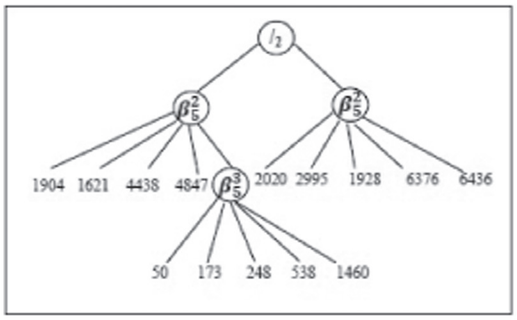
Leukemia is the scientific noun of the blood cancer. The leukemia dataset consists of 72 samples obtained from leukemia patients. Each contains 7129 features [30]. There are 47 patients suffering from acute myeloid leukemia (AML) and the rest suffering from acute lymphoblastic leukemia (ALL). The dataset was divided into two subsets of samples: a training set contains 38 samples (27 ALL and 11 AML) and a testing set contains 34 samples (20 ALL and 14 AML).

4.1.3. Diffuse large b-cell lymphoma (DLL Lymphoma)

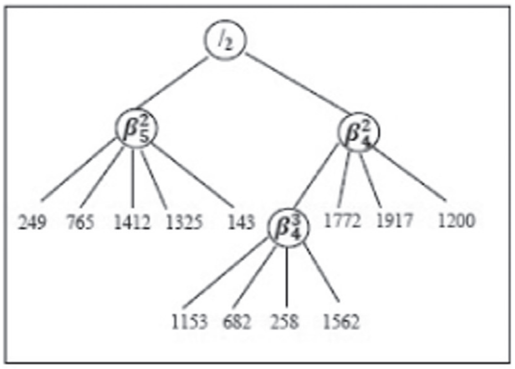
DLBC Lymphoma is a type of cancer affecting the cells (B white blood cell) which produce antibodies in the human blood. It is characterized by the aggressive behavior in the way that the malicious cells spread widely in the different organs. The DLBC dataset consists of 27 instances for training and 20 instances for testing [31]. Each instance/sample is formed by 4026 features and belongs to one of these classes: germinal (24 samples) or activated (23 samples).

4.1.4. Lung cancer dataset

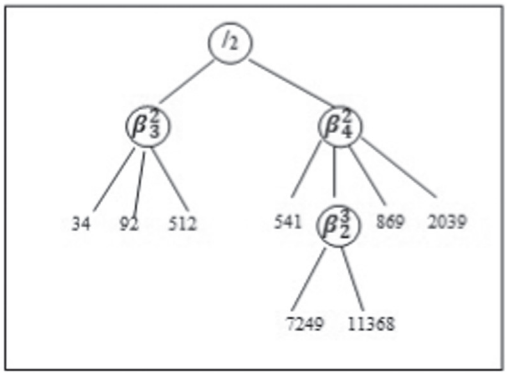
According to last statistics, the Lung cancer is classified as the second most common cancer in both men and women. The



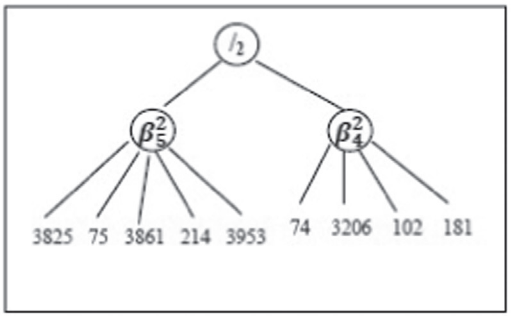
(1)The optimal solution of PMA_FNT with Leukemia dataset



(2)The optimal solution of PMA_FNT with Colon cancer dataset



(3)The optimal solution of PMA_FNT with Lung cancer dataset



(4)The optimal solution of PMA_FNT with Lymphoma dataset

Fig. 9. Representation of the PMA_FNT solutions. (1) The optimal solution of PMA_FNT with Leukemia dataset. (2) The optimal solution of PMA_FNT with Colon cancer dataset. (3) The optimal solution of PMA_FNT with Lung cancer dataset. (4) The optimal solution of PMA_FNT with Lymphoma dataset.

surviving patients' rates vary depending on early stage diagnosis. The lung cancer dataset contains two classes; the malignant pleural mesothelioma (MPM) and adenocarcinoma (ADCA) of the lung. They are described through 12 533 genes (features) and 181 tissue samples [32]. This dataset is divided into training set containing 32 samples (16 MPM and 16 ADCA) and testing set containing 149 samples (15 MPM and 134 ADCA).

4.2. Feature selection

Recently, the data management has become more difficult due to the immense data growth in terms of number of instances and number of features. Many machine learning algorithms suffer from this enormity which degrades their learning performance. In many applications such as classification, image retrieval and genome projects, high dimensional data could not be free from some irrelevant and redundant information that misleads the system [33]. Therefore, data preparation known as feature selection is the key part for successful learning.

Feature selection (FS) is a process of searching the best subset of attributes in the original dataset. It reduces the feature space according to the method evaluation criteria. The objective of such FS algorithm is to find the best combinations of features instead of working on all features of the dataset. The removal of redundant and misleading data might reduce the over fitting of the training time and improve accuracy. In our experiments, we benefited from the decision tree algorithm (J4.8 version), implemented in Waikato Environment for Knowledge Analysis (WEKA) [34], to prepare our datasets presented previously. After applying the feature selection phase, the number of features was reduced . The list of the selected features is shown in Table 1.

4.3. Results and discussion

In the PMA_FNT model, the used tree degree is between 2 and 6 as respectively minimum and maximum. Table 2 illustrates the initialized parameter values in the experimentation.

Our model was applied on four datasets using the training data for training the FBBFNTs and testing dataset for testing the best model.

Table 3 contains a set of performance measures to evaluate our model. Relying on the confusion matrix, many performance measures, such as accuracy, were used to evaluate the classifier. The confusion matrix visualizes the number of the correct identification of positive samples (TP True Positive) and negative samples (TN True Negative) and the number of misclassified cases of positive samples (FP False Positive) and negative samples (FN False Negative) [35].

In Table 3, we have presented training and testing accuracy mean rates generated after 20 runs. The accuracy rates are always over 99.3 percent which prove that the proposed method is effective in the classification for all presented datasets. Moreover, we have presented the confusion matrix of the worst result during these runs. The fact that the misclassification rate (FP and FN) is low even in the worst case proves the stability and the good performance of our model. Along with accuracy, the Root Mean Squared Error (RMSE) is considered to prove the good performance of our model.

In order to make these results more meaningful, a comparative study between the PMA_FNT model and other classifiers from the literature was made. Four bar chart graphics were disposed to illustrate the different accuracy rates (see Fig. 7).

In the comparison study, all works used the same datasets presented previously. In the different tests, we compared our model with other kind of classifiers like SVM [36–39], K-NN [40,41], BoostCART [42,43] and Fuzzy system HFBFS [44]. It resulted in a satisfactory improvement with range around 3% with Leukemia dataset, between 4% and 6% with Lymphoma dataset, between 0.3% and 8% with lung cancer dataset, and more than 10% with Colon cancer dataset. These results report that the evolved ANN is a good competitor of the other categories of classifiers.

Our goal is not only to prove the effectiveness of the PMA_FNT model as a classifier, but also to show the good effect of the multi-agent training process of this model on the results' accuracy. As a result, we tried to compare our model with the FNT model and the FBBFNT model. They were evolving radial and beta FNTs, respectively. The PMA_FNT model shows a light improvement of the classification accuracy, but we had other performance measurements used in this comparison. Next to the classification accuracy, the time complexity computed by the Number of Function Evaluations (NFEs) and the structure complexity of the Neural Network were adapted in this study.

Regarding the time complexity, Fig. 8 illustrates the operated time (NFEs) used by two methods (FBBFNT and PMA_FNT) during the classification of the four datasets (leukemia, colon cancer, lymphoma, and lung cancer). With a different test, our model could reach better results in less time (NFEs). Compared with the FBBFNT model, PMA_FNT reduces the NFEs from 60 355 to 43 717 with Leukemia dataset, from 43 007 to 24 161 with Lymphoma dataset, from 55 421 to 32 108 with colon cancer dataset, and from 32 108 to 10 027 with Lung cancer dataset (see Fig. 8). These impressive results were explained by the reduction of the global iteration/round number. It was due to the cooperative work of the interacted agents to obtain the optimum through negotiation. The parallel training and the use of many agents optimizing a set of feasible solutions gave the chance to the model to accelerate the process and explore better solutions. The better formed solutions are defined as good classifiers with better structure complexity (see Fig. 9).The presented solutions of our model are characterized by the limited number of connections, inputs/features, and the low degree of the tree.

5. Conclusion

In this paper, we proposed a parallelized learning process for the Flexible Beta basis Neural Network using a multi-agent architecture. This system is called the Pareto Multi-Agent flexible Neural Tree (PMA_FNT). It looks for the optimal neural network respecting two main objectives; the accuracy and the structure complexity. For that, PMA_FNT applies a multi-objective optimization based on the Pareto dominance. It distributes the learning process to a Structure agent and a variable set of Parameter agents. Two evolutionary algorithms are used for the optimization of the ANN; the Pareto-dominance Extended Genetic Programming for the ANN structure optimization PEGP and the Pareto Multi-dimensional Particle Swarm Optimization PMD_PSO for the ANN parameters optimization. These agents can work in parallel with exchanging messages and information. In fact, our system operates through a communication process between different agents managed by the negotiation protocol and the agent strategy. It relies on the ADR and ATV factors to evaluate the agent performance and ensure a competitive environment.

The system was evaluated using three real datasets for classification which are Leukemia, Colon cancer and DLBC Lymphoma datasets. Its comparison with other classifiers from the literature proved its higher performance, efficiency and speed.

Acknowledgement

The authors would like to acknowledge the financial support to this work by grants from the General Direction of Scientific Research and Technological Renovation (DGRSRT), Tunisia, under the ARUB program 01/UR/11/02. Ajith Abraham acknowledges the support from the framework of the IT4 Innovations Center of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Program 'Research and Development for Innovations' funded by Structural Funds of the European Union and state budget of the Czech Republic.

References

[1] M. Wooldridge, An Introduction to Multiagent Systems, John Wiley & Sons LTD, UK, 2009.
[2] S. Bouaziz, H. Dhahri, A.M. Alimi, A. Abraham, A hybrid learning algorithm for evolving flexible beta basis function neural tree model, Neurocomputing 117 (2013) 107–117.
[3] Y. Chen, A. Abraham, Tree-Structure based Hybrid Computational Intelligence: Theoretical Foundations and Applications & Business Media, vol. 2, Springer Science, 2009.
[4] P.K. Simpson, Fuzzy min-max neural networks. i. classification, IEEE Trans. Neural Netw. 3 (5) (1992) 776–786.
[5] M. Ammar, S. Bouaziz, A.M. Alimi, A. Abraham, Negotiation process for bi-objective multi-agent flexible neural tree model, in: 2015 International Joint Conference on Neural Networks (IJCNN), IEEE, 2015, pp. 1–9.
[6] S. Wang, F.-L. Chung, J. Wang, J. Wu, A fast learning method for feedforward neural networks, Neurocomputing 149 (2015) 295–307.
[7] W. Yu, F. Zhuang, Q. He, Z. Shi, Learning deep representations via extreme learning machines, Neurocomputing 149 (2015) 308–315.
[8] G.A. Montazer, D. Giveki, An improved radial basis function neural network for object image retrieval, Neurocomputing 168 (2015) 221–233.
[9] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International of First Conference on Neural Networks, 1995.
[10] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 1975.
[11] M. Ammar, S. Bouaziz, A. M. Alimi, A. Abraham, Hybrid harmony search algorithm for global optimization, in: 2013 World Congress on Nature and Biologically Inspired Computing (NaBIC), IEEE, 2013, pp. 69–75.
[12] S. Bouaziz, H. Dhahri, A.M. Alimi, A. Abraham, Evolving flexible beta basis function neural tree using extended genetic programming & hybrid artificial bee colony, Appl. Soft Comput. (2016).
[13] G. Weiss, Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.
[14] H. Dhahri, A.M. Alimi, A. Abraham, Hierarchical multi-dimensional differential evolution for the design of beta basis function neural network, Neurocomputing 97 (2012) 131–140.
[15] P. Farber, K. Asanovic, Parallel neural network training on multi-spert, in: 1997 3rd International Conference on Algorithms and Architectures for Parallel Processing, ICAPP 97, IEEE, 1997, pp. 659–666.
[16] N. Mache, P. Levi, Parallel Neural Network Training and Cross Validation on a Cray t3e System and Application to Splice Site Prediction in Human DNA, Technical Report, Technical report, Institute of Parallel and Distributed High Performance Systems, Stuffgart, Germany, 1995.
[17] N.R. Suri, D. Deodhare, P. Nagabhushan, Parallel Levenberg–Marquardt-based neural network training on linux clusters—a case study., in: ICVGIP, 2002.
[18] G. Dahl, A. McAvinney, T. Newhall, et al., Parallelizing neural network training for cluster systems, in: Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, ACTA Press, 2008, pp. 220–225.
[19] A. Quteishat, C.P. Lim, J. Tweedale, L.C. Jain, A neural network-based multi-agent classifier system, Neurocomputing 72 (7) (2009) 1639–1647.
[20] Z.-S. Zhao, X. Feng, Y.-y. Lin, F. Wei, S.-K. Wang, T.-L. Xiao, M.-Y. Cao, Z.-G. Hou, Evolved neural network ensemble by multiple heterogeneous swarm intelligence, Neurocomputing 149 (2015) 29–38.
[21] A.M. Alimi, The beta fuzzy system: approximation of standard membership functions, Proc. 17eme J. Tunis. d'Electrotech. d'Autom.: JTEA 97 (1997) 108–112.

[22] S. Bouaziz, A.M. Alimi, A. Abraham, Universal approximation propriety of flexible beta basis function neural tree, in: 2014 International Joint Conference on Neural Networks (IJCNN), IEEE, 2014, pp. 573–580.

[23] J.R. Koza, Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University, Department of Computer Science, 1990.

[24] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[25] S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization, Neural Netw. 22 (10) (2009) 1448–1462.

[26] S. Fatima, I. Rahwan, Negotiation and bargaining, in: Multiagent Systems, 2013, p. 143.

[27] F. Lopes, M. Wooldridge, A.Q. Novais, Negotiation among autonomous computational agents: principles, analysis and challenges, Artif. Intell. Rev. 29 (1) (2008) 1–44.

[28] R.G. Smith, R. Davis, Frameworks for cooperation in distributed problem solving, IEEE Trans. Syst. Man Cybern. 11 (1) (1981) 61–70.

[29] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (5439) (1999) 531–537.

[30] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, A.J. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Proc. Natl. Acad. Sci. 96 (12) (1999) 6745–6750.

[31] I.S. Lossos, A.A. Alizadeh, M.B. Eisen, W.C. Chan, P.O. Brown, D. Botstein, L.M. Staudt, R. Levy, Ongoing immunoglobulin somatic mutation in germinal center b cell-like but not in activated b cell-like diffuse large cell lymphomas, Proc. Natl. Acad. Sci. 97 (18) (2000) 10209–10213.

[32] G.J. Gordon, R.V. Jensen, L.-L. Hsiao, S.R. Gullans, J.E. Blumenstock, S. Ramaswamy, W.G. Richards, D.J. Sugarbaker, R. Bueno, Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma, Cancer Res. 62 (17) (2002) 4963–4967.

[33] L. Yu, H. Liu, Feature selection for high-dimensional data: a fast correlation-based filter solution, in: ICML, vol. 3, 2003, pp. 856–863.

[34] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.

[35] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013) 27–38.

[36] X.-w. Chen, Gene selection for cancer classification using bootstrapped genetic algorithms and support vector machines, in: Proceedings of the 2003 IEEE Bioinformatics Conference, 2003, CSB 2003, IEEE, 2003, pp. 504–505.

[37] S.-B. Cho, H.-H. Won, Machine learning in dna microarray analysis for cancer classification, in: Proceedings of the First Asia–Pacific Bioinformatics Conference on Bioinformatics 2003, vol. 19, Australian Computer Society, Inc., 2003, pp. 189–198.

[38] R. Luque-Baena, D. Urda, J. Subirats, L. Franco, J. Jerez, Analysis of cancer microarray data using constructive neural networks and genetic algorithms, in: Proceedings of the IWBBIO, International Work-Conference on Bioinformatics and Biomedical Engineering, 2013, pp. 55–63.

[39] A.H. Chen, C.H. Lin, C.H. Cheng, New approaches to improve the performance of disease classification using nested-random forest and nested-support vector machine classifiers, Cancer 2 (10509) 102.

[40] A.M. Mahmoud, B.A. Maher, A hybrid reduction approach for enhancing cancer classification of microarray data, Int. J. Adv. Res. Artif. Intell. 3(10) (2014).

[41] L. Li, T.A. Darden, C. Weingberg, A. Levine, L.G. Pedersen, Gene assessment and sample classification for gene expression data using a genetic algorithm/k-nearest neighbor method, Comb. Chem. High Throughput Screen. 4 (8) (2001) 727–739.

[42] S. Dudoit, J. Fridlyand, T.P. Speed, Comparison of discrimination methods for the classification of tumors using gene expression data, J. Am. Stat. Assoc. 97 (457) (2002) 77–87.

[43] S. Pang, I. Havukkala, Y. Hu, N. Kasabov, Classification consistency analysis for bootstrapping gene selection, Neural Comput. Appl. 16 (6) (2007) 527–539.

[44] Y. Jarraya, S. Bouaziz, A.M. Alimi, A. Abraham, Evolutionary multi-objective optimization for evolving hierarchical fuzzy system, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 3163–3170.

Marwa Ammar was born in Sfax (Tunisia) in 1987. She graduated in Computer Engineering 2011. She is currently working toward the Ph.D degree with the University of Sfax. Her research interest includes computational intelligence: artificial neural network, evolutionary computation, multi-agent system.

Souhir Bouaziz was born in Sfax (Tunisia) in 1984. She graduated in Computer Engineering 2008 and obtained a Ph.D. degree in Computing System Engineering in 2013 at the University of Sfax. She is currently an Assistant Professor in the University of Gabes. Her research interest includes computational intelligence: neural network, evolutionary computation, swarm intelligence.

Adel M. Alimi was born in Sfax (Tunisia) in 1966. He graduated in Electrical Engineering 1990, obtained a Ph.D. and then an HDR both in Electrical & Computer Engineering in 1995 and 2000 respectively. He is now a Professor in Electrical & Computer Engineering at the University of Sfax. His research interest includes applications of intelligent methods (neural networks, fuzzy logic, evolutionary algorithms) to pattern recognition, robotic systems, vision systems, and industrial processes. He focuses his research on intelligent pattern recognition, learning, analysis and intelligent control of large scale complex systems. He is an Associate Editor and Member of the editorial board of many international scientific journals (e.g. "Pattern Recognition Letters", "Neurocomputing", "Neural Processing Letters", "International Journal of Image and Graphics", "Neural Computing and Applications", "International Journal of Robotics and Automation", "International Journal of Systems Science", etc.). He was a Guest Editor of several special issues of international journals (e.g. Fuzzy Sets & Systems, Soft Computing, Journal of Decision Systems, Integrated Computer Aided Engineering, Systems Analysis Modelling and Simulations). He was the General Chairman of the International Conference on Machine Intelligence ACIDCA-ICMI'2005 & 2000. He is an IEEE senior member and member of IAPR, INNS and PRS. He is the 2009–2010 IEEE Tunisia Section Treasurer, the 2009–2010 IEEE Computational Intelligence Society Tunisia Chapter Chair, the 2011 IEEE Sfax Subsection, the 2010–2011 IEEE Computer Society Tunisia Chair, the 2011 IEEE Systems, Man, and Cybernetics Tunisia Chapter, the SMCS corresponding member of the IEEE Committee on Earth Observation, and the IEEE Counselor of the ENIS Student Branch.

Ajith Abraham received the Ph.D. degree in Computer Science from Monash University, Melbourne, Australia. He is currently the Director of Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA, which has members from more than 85 countries. He has a worldwide academic and industrial experience of over 20 years. He works in a multi-disciplinary environment involving machine intelligence, network security, various aspects of networks, e-commerce, Web intelligence, Web services, computational grids, data mining, and their applications to various real-world problems. He has numerous publications/citations (h-index 40) and has also given more than 50 plenary lectures and conference tutorials in these areas. Since 2008, he is the Chair of IEEE Systems Man and Cybernetics Society Technical Committee on Soft Computing and a Distinguished Lecturer of IEEE Computer Society representing Europe (since 2011). Dr. Abraham is a Senior Member of the IEEE, the Institution of Engineering and Technology (UK) and the Institution of Engineers Australia (Australia), etc. He is the founder of several IEEE sponsored annual conferences, which are now annual events. More information at: ⟨http://www.softcomputing.net⟩.