A Hybrid Hopfield Network-Imperialist Competitive Algorithm for Solving the Satisfiability Problems

Marjan Abdechiri and Mohammad Reza Meybodi

***Abstract*—The novel Imperialist Competitive Algorithm (ICA) was recently introduced has a good performance in some optimization problems. The ICA inspired by socio-political process of imperialistic competition of human being in the real world. In this paper is proposed two algorithm for Solving SAT problems. Firstly, ICA algorithm is used to solve SAT problem that this algorithm is called ICALR. The SAT problem is formulated to Lagrange multipliers, which repreent the weights of the clauses of SAT. Secondly, this paper presents a hybrid Hopfield network (HNN)-Imperialist Competitive Algorithm (ICA) to solve the SAT problem. In this proposed algorithm, Hopfield neural network (HNN) manages the problem's constraints and ICA algorithm searches for high quality solutions and minimum cost. ICALR and HNNICA algorithm are used for solving Lagrange multipliers. Some famous benchmark sets containing instances from SAT problems from various domains used to test the ICALR and HNNICA performance. Also we present a detailed comparative analysis of the proposed algorithms' performance. Simulation results show HNNICA strategy can improve the performance of the HNN algorithm significantly. Experimental results show that the HNNICA finds solutions faster than existing methods.**

***Index Terms*—Imperialist competitive algorithm, hopfield neural network, sat problem, lagrange multipliers.**

I. Introduction

Many EAs have been proposed for the problems of global optimization. So far, different evolutionary algorithms have been proposed for optimization. Recently, a new algorithm has been proposed by Atashpaz-Gargari and Lucas [1] which has inspired not natural phenomenon, but of course from a socio-human from phenomenon. This algorithm has looked at imperialism process as a stage of human's socio-political evolution. The Imperialist Competitive Algorithm makes relation between humans and social sciences on one hand, and technical and mathematical sciences on the other hand, having a completely new viewpoint about the optimization topic. In the ICA algorithm, the colonies move towards the imperialist country with a random radius of movement. In [2] CICA algorithm has been proposed that improved performance of ICA algorithm by thechaotic maps are used to adapt the angle of colonies movement towards imperialist's position to enhance the escaping capability from a local optima trap. The ICA algorithm is used for Neural Network Learning based on Chaotic Imperialist Competitive Algorithm [3].

Manuscript received August 23, 2012; revised September 21, 2012

Marjan Abdechiri is with Electronic, Computer and IT Department, Qazvin Azad University, Qazvin, Iran (e-mail: marjan.abdechiri@qiau.ac.ir)

Mohammad Reza Meybodi is with Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran (e-mail: mmeybodi@aut.ac.ir)

Binary HNN is a Hopfield neural network used for solving constraints in optimization problems [4]. In [5] the HNN algorithm is hybridized with genetic algorithm. In [6] the HNN algorithm is hybridized with simulated annealing.

The aim of this paper is twofold: First we apply ICA algorithm to solving SAT problem and second we present a novel hybrid Hopfield network and ICA for solving SAT problems that constraint are managed by the HNN and the quality of the solution obtained is improved by the ICA algorithm.

The proposed algorithm (HNNICA) has a good performance for solving SAT problems. The HNN algorithm reduces the search space of the ICA algorithm. The performance of HNNICA has been evaluated in several SAT problems and it compared with the ICALR proposed and LPPH with very good results in all test instances considered. The empirical results obtained using the benchmarks indicate that the speed of convergence and the quality of solutions are better than LPPH. The HNNICA and ICALR algorithms have been tested on randomly generated problems and some classes of problems from the DIMACS test set.

The rest of this paper organized as follows: Section 2,provides an introduction of SAT problem. In section 3, Imperialist Competitive Algorithms are reviewed. In section 4, two new algorithms proposed for solving SAT. In section 5, is devoted to simulation results and finally section 6, conclusion and related work the paper.

II. Satisfiability Problem

The Satisfiability (SAT) problems belong to an important class of discrete constraint satisfaction problems (CSP). The SAT problem is a set of m clauses $\{C_1, C_2, \dots, C_m\}$ on n variables $x = (x_1, x_2, \dots, x_n)$, $x_i \in \{0,1\}$ and a Boolean formula in conjunctive normal form (CNF).

$$C_1 \wedge C_2 \wedge \dots \wedge C_m \qquad (1)$$

A clause is a combination of m literals where a literal is a Boolean variable x_i or its negation $\bar{x}_i$. If the number of literals is n for all clauses, then SAT is referred to as n-SAT. The goal of the SAT problem is to determine whether there exists an assignment of truth values to variables that makes the CNF formula satisfiable. The propositional satisfiability (or SAT) problem is one of researched problems in computer science and artificial intelligence, mathematics, machine vision, robotics and computer-aided manufacturing. In this paper, we transform the fitness function to Lagrange Multipliers representation. So objective in continuous space may "smooth out" some infeasible solutions, leading to a smaller number of local minima explored.

$$min_{y\in E^n} f(y) = \sum_{i=1}^{m} c_i(y), \quad (2)$$

where

$$c_i(y) = \prod_{j=1}^{n} q_{i,j}(y_j), \quad (3)$$

$$q_{i,j}(y_j) = \begin{cases} y_j & if x_j inc_i \\ 1 - y_j & if \overline{x_j} inc_i \\ 1 & otherwise \end{cases} \quad (4)$$

fitness function SAT is

$$(SAT) \qquad find\, x \quad (5)$$

such that x satisfies $c_r, r = 1, 2, \dots, m x \in \{0,1\}^n$.
Fitness function CONSAT is

$$\begin{gathered}(\text{CONSAT}) \quad \text{findx} \\ \text{such that } h_r(x) = 0, \quad r = 1,2,\dots,m \\ x \in [0,1]^n. \end{gathered} \quad (6)$$

where

$$h_r(x) = \prod_{i=1}^{n} g_{ir}(x) \quad (7)$$

The above problem can be reformulated using Lagrange multipliers into the following unconstrained problem Lagrange function for SAT is defined as follow:

$$F(x, w) = \sum_{r=1}^{m} w_r h_r(x),$$

$$x \in [0,1]^n, w \in (0, \infty)^m \quad (8)$$

The last decade, several algorithms have been developed for solving the SAT problem. These algorithms divided into two main classes: complete (Systematic) and incomplete algorithms (non-systematic). An algorithm is *complete,* if return a solution for a problem if one exists, and prove it unsolvable otherwise. The most efficient complete algorithms are based on the Davis-Putnam-Loveland procedure [9]. This algorithm is a complete, backtracking-based algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form. Several algorithmic variations of the Davis-Putnam procedure have been proposed [10].

The SAT problems are hard to solve using complete algorithms. One way to overcome for this problem is Local search algorithms [11], [12], [13] and evolutionary algorithms [14]-[18].These include techniques Such as Simulated Annealing [19], WSAT algorithm [20], GSAT [21], Tabu Search [22] and Learning automata [23].

The GSAT starts with a randomly generated truth assignment. It then changes the variable assignment that leads to the largest increase in the total number of clause satisfied. GSAT is greedy because it always tries to increase the number of true clauses.

In [24] is introduce the Lagrange programming neural network methods for SAT problem and showed that the LPPH algorithm outperforms the DPL and the BM algorithms. In [25] analyzed the stability and the local convergence properties LPPH algorithm.

Dynamical differential equations for LPPH are defined as follows: LPPH algorithm is a lagrangian programming neural network with polarized high-order connections. In this algorithm neuron i corresponds to variable x_i. The potential of neuron i is u_i. Each neuronhas a polarity for each connection. If literal in clause is negative so polarity is negative. The input to neuron i via connection r is $\prod_{j\neq i} g_{jr}(x)$ or $\left(-\prod_{j\neq i} g_{jr}(x)\right)$ and dynamic differential equations for LPPH are

$$\frac{du_i}{dt} = -\sum_{r=1}^{m} w_r \frac{\partial h_r(x)}{\partial x_i}, \qquad i = 1, 2, \dots, n. \quad (9)$$

$$x_i = \frac{1}{1+e^{-u_i}}, \qquad i = 1, 2, \dots, n. \quad (10)$$

$$\frac{dw_r}{dt} = h_r(x), \qquad r = 1, 2, \dots, m. \quad (11)$$

$$\frac{dx_i}{dt} = -x_i(1 - x_i)\frac{\delta F(x, w)}{\delta x_i} = x_i(1 - x_i)\sum_{i=1}^{m} w_r \frac{\partial h_r(x)}{\partial x_i} \qquad i = 1,2,\dots,n \quad (12)$$

$$\frac{dw_r}{dt} = -\alpha w_r + \frac{\delta F(x, w)}{\delta w_r} = -\alpha w_r + h_r(x), \qquad r = 1,2,\dots,m \quad (13)$$

Here $F(x, w) \geq 0$ and $F(x, w) = 0 iff x$ is a solution of the CONSAT. That α is the attenuation coefficient, which influences the performance of the LPPH. Each weight $\mathrm{w_r}$ increases according to the degree of unsatisfiability of clause C_r. This causes changes in the energy landscape of the Lagrangian function. The values of the variables change using the gradient descent. The instances Benchmark of satisfiability problems are used for comparing results are shown in Tables I, II.

TABLE I: CNF'S USED IN THE EXPERIMENT

Name	Variable	Clause	
EXPD1	50	100	3-SAT. Unique solution
EXPD2	100	430	
EXPF1	319	941	3-SAT
EXPF2	318	940	
EXPH1	100	762	Test pattern generation of combinational circuits
EXPH2	225	2799	
EXPU1	50	120	Randomly generated 3-SAT unique solution
EXPU2	50	130	

TABLE II: BENCHMARK FOR SIMULATION

Name	Instances	Variables	Clauses
Uf50-218/uuf50-218	2*1000	50	218
Uf75-325/uuf75-325	2*100	75	325
Uf100-430/uuf30-430	2*1000	100	430
Uf125-538/uuf125-538	2*100	125	538
Uf150-645/uuf150-645	2*100	150	645
Uf175-753/uuf175-753	2*100	175	753
Uf200-860/uuf200-860	2*100	200	860
Uf225-960/uuf225-960	2*100	225	960
Uf250-1065/uuf250-1065	2*100	250	1065

III. Introduction of Imperialist Competitive Algorithm (ICA)

Imperialist Competitive Algorithm (ICA) is a new evolutionary algorithm in the Evolutionary Computation field based on the human's socio-political evolution. The algorithm starts with an initial random population called countries. Some of the best countries in the population selected to be the imperialists and the rest form the colonies of these imperialists. In an N dimensional optimization problem, a country is $1 \times N$ array. This array defined as below

$$country = [p_1, p_2, \dots, p_N] \tag{14}$$

The cost of a country is found by evaluating the cost function (f)at the variables ($p_1, p_2, p_3, \dots, p_N$). Then

$$c_i = f(country_i) = f(p_{i1}, p_{i2}, \dots, p_{iN}) \tag{15}$$

The algorithm starts with N initial countries and the N_{imp} best of them (countries with minimum cost) chosen as the imperialists. The remaining countries are colonies that each belong to an empire. The initial colonies belong to imperialists in convenience with their powers. To distribute the colonies among imperialist proportionally, the normalized cost of an imperialist is defined as follow

$$C_n = max_i\, c_i - c_n \tag{16}$$

where, $cost_n$ is the cost of nth imperialist and C_n is its normalized cost. Each imperialist that has more cost value, will have less normalized cost value. Having the normalized cost, the power of each imperialist is calculated as below and based on that the colonies distributed among the imperialist countries.

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \tag{17}$$

On the other hand, the normalized power of an imperialist is assessed by its colonies. Then, the initial number of colonies of an empire will be

$$NC_n = rand\{p_n . (N_{col})\} \tag{18}$$

where, NC_n is initial number of colonies of nth empire and N_{col} is the number of all colonies.To distribute the colonies among imperialist, NC_n of the colonies is selected randomly and assigned to their imperialist. The imperialist countries absorb the colonies towards themselves using the absorption policy. The absorption policy shown in Fig. 1 makes the main core of this algorithm and causes the countries move towards to their minimum optima. The imperialists absorb these colonies towards themselves with respect to their power that described in Eq. 19. The total power of each imperialist is determined by the power of its both parts, the empire power plus percents of its average colonies power.

$$TC_n = cost(imperialist_n) + \xi\text{mean}\{\text{cost}(\text{coloniesofempire}_\text{n}\} \tag{19}$$

where TC_n is the total cost of the nth empire and ξ is a positive number which is considered to be less than one.

$$x \sim U(0, \beta \times d) \tag{20}$$

In the absorption policy, the colony moves towards the imperialist by x unit. The direction of movement is the vector from colony to imperialist, as shown in Fig. 1. In this figure, the distance between the imperialist and colony shown by d and x is a random variable with uniform distribution. Where β is greater than 1 and is near to 2. So, a proper choice can be $\beta = 2$. In our implementation γ is $\pi/_4\,(Rad)$ respectively.

$$\theta \sim U(-\gamma, \gamma) \tag{21}$$

In ICA algorithm, to search different points around the imperialist, a random amount of deviation is added to the direction of colony movement towards the imperialist. In Fig. 1, this deflection angle is shown as θ, which israndomly chosen and with an uniform distribution. While moving toward the imperialist countries, a colony may reach to a better position, so the colony position changes according to position of the imperialist.

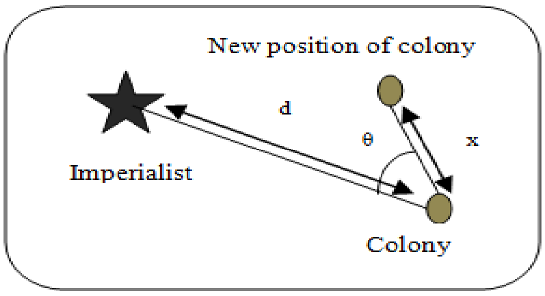


Fig. 1. Moving colonies toward their imperialist

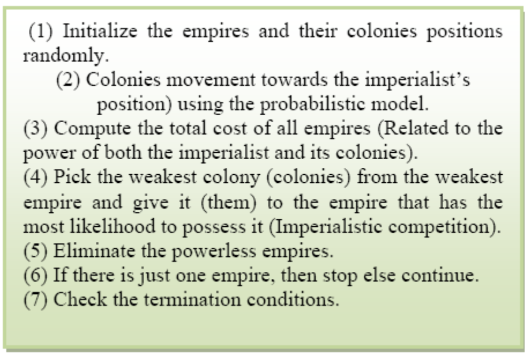
In this algorithm, the imperialistic competition has an important role. During the imperialistic competition, the weak empires will lose their power and their colonies. To model this competition, firstly we calculate the probability of possessing all the colonies by each empire considering the total cost of empire.

$$NTC_n = max_i\{TC_i\} - TC_n \tag{22}$$

where, TC_n is the total cost of nth empire and NTC_n is the normalized total cost of nth empire. Having the normalized total cost, the possession probability of each empire is calculated as below

$$p_{p_n} = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \tag{23}$$

After a while all the empires except the most powerful one will collapse and all the colonies will be under the control of this unique empire. The sequence of ICA algorithm is shown in Fig. 2.



(1) Initialize the empires and their colonies positions randomly.
(2) Colonies movement towards the imperialist's position) using the probabilistic model.
(3) Compute the total cost of all empires (Related to the power of both the imperialist and its colonies).
(4) Pick the weakest colony (colonies) from the weakest empire and give it (them) to the empire that has the most likelihood to possess it (Imperialistic competition).
(5) Eliminate the powerless empires.
(6) If there is just one empire, then stop else continue.
(7) Check the termination conditions.

Fig. 2.The ICA Algorithm

IV. The Proposed Algorithms for Solving SAT Problem

In this section, two new algorithms to solve the SAT areproposed. Firstly, ICA algorithm is used for finding the Lagrange multipliers in SAT problem. Secondly, we improve ICALR with Hopfield Neural Network.

A. ICAR Algorithm

In this section, the ICALR to solve the SAT is explained. The ICA approach is used to search for the Lagrange multipliers w that represent the weights of the clauses of the SAT. Firstly, the solving SAT problem is performed using ICA algorithm. This algorithm is called ICALR. The ICALR is used for solving Lagrange Multipliers of SAT problem. Lagrange Multipliers and the variables of SAT problem are solutions and any individual is introduced with variables and multipliers are shown with Matrix $w$and x. The number of Lagrange Multipliers is m.

$$w = \begin{matrix} w_{11} w_{12} \ldots\ldots w_{1m} \\ w_{21} w_{22} \ldots\ldots w_{2m} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ w_{p1} w_{p2} \ldots\ldots w_{pm} \end{matrix} \qquad (24)$$

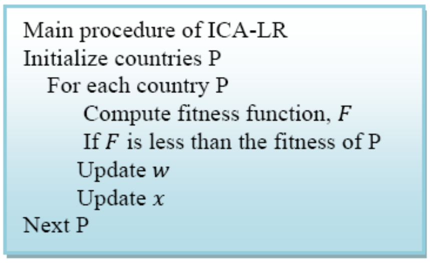
The number of variables of is n.

$$X = \begin{matrix} x_{11} x_{12} \ldots\ldots x_{1n} \\ x_{21} x_{22} \ldots\ldots x_{2n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_{p1} x_{p2} \ldots\ldots x_{pn} \end{matrix} \qquad (25)$$

A country, one row in matrix w, represents a set of the Lagrange multipliers corresponding to each clause. Each colony has a structure with two matrices w, x. The fitness of a country is defined as

$$F(x,w) = \sum_{r=1}^{m} w_r h_r(x),$$
$$x \in [0,1]^n, w \in (0,\infty)^m \ . (26)$$

In ICA approach, the fitness of a country is interpreted as the worth of its location in the search space. The matrices w and x are updated in each decade. The pseudo code for the proposed method is shown below:



```
Main procedure of ICA-LR
Initialize countries P
  For each country P
      Compute fitness function, $F$
      If $F$ is less than the fitness of P
     Update $w$
     Update $x$
Next P
```

Fig. 3.The ICALR algorithm

B. HNNICA Algorithm

In This section, we propose to combine the HNN with ICA algorithm to solving SAT problem. The HNNICA algorithm is formed using the HNN and the ICA. In the proposed algorithm, HNN runs before calculating the fitness function for each individual in the ICA. The HNN reduces the search space of ICA algorithm and HNN discards some solutions that are not feasible for SAT problem. In the HNNICA, HNN is for local search and ICA is good for global searchability.

The HNNICA is proposed for solving Lagrange multipliers. In Fig. 4, the Diagramof the HNN algorithm is shown.

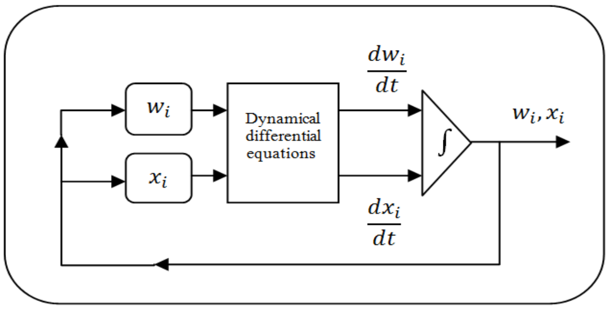


Fig. 4. Diagram of HNN for solving SAT problem

The procedure of the proposed algorithm is shown as follows:

Step1: The SAT problem is formulated to a Lagrange function that this function is unconstraint and continues.

Step2: Initialize countries of ICA (random).

Step3: For every individual run the HNN to obtain a feasible P. (In Fig. 5, Flowchart of HNN for solving the SAT problem is shown.)

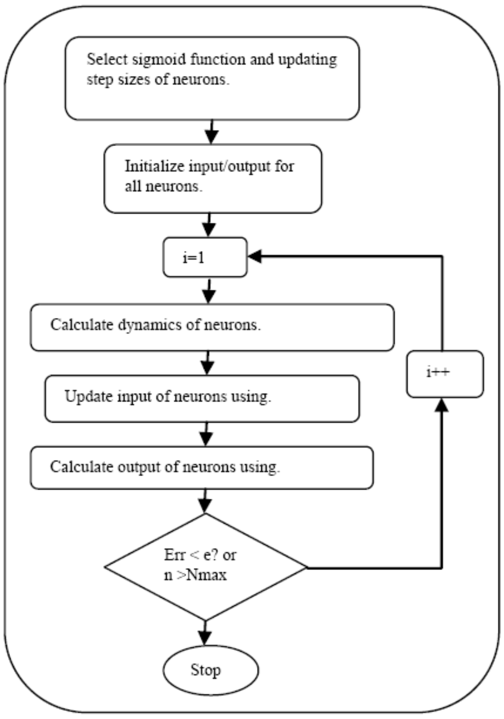

Select sigmoid function and updating step sizes of neurons.
Initialize input/output for all neurons.
i=1
Calculate dynamics of neurons.
Update input of neurons using.
Calculate output of neurons using.
i++
Err < e? or n >Nmax
Stop

Fig. 5. Flowchart of HNN for solving SAT problem in HNNICA algorithm

Step4: Calculate the fitness value of the individual.

Step5: Create new individual.

Step6: Moving colonies toward imperialists and continues.

Step 7: While (Number of decades).

In Hopfield is described as follows: the output of neurons represents value of lagrange multipliers and variables of SAT. The weights are chosen such that an energy function can be defined that it is lagrangian function and lowest energy state is best solution. Hopfield evolve energy function in time to decrease monotonically until a stable steady state is reached. This steady state is corresponded to the optimal solution. First SAT encoded to Lagrangian function and secondly an energy function and the corresponding weights have to be determined. In Fig. 5 show diagram of Hopfield network is used in HNNICA. In HNNICA runs HNN algorithm for 20 iterations. InFigs. 6,7 details of implementation HNNICA algorithm are shown.

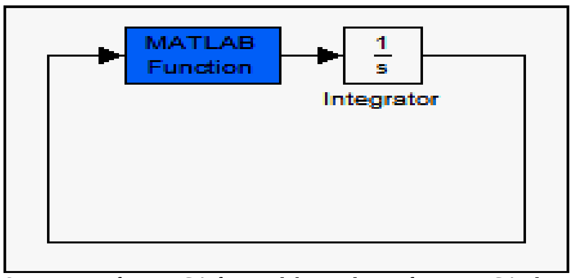
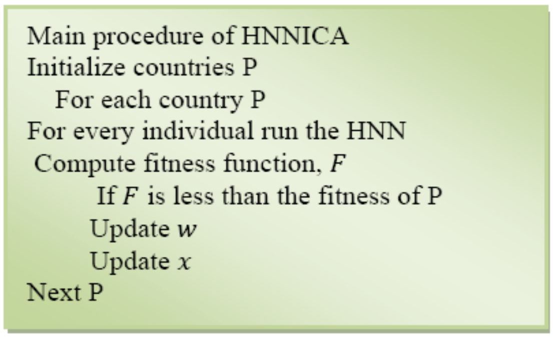


Fig. 6. Diagram of HNNICA for Matlab simulation for HNNICA algorithm



```
Main procedure of HNNICA
Initialize countries P
    For each country P
For every individual run the HNN
  Compute fitness function, $F$
        If $F$ is less than the fitness of P
        Update $w$
        Update $x$
Next P
```

Fig. 7.HNNICA algorithm

V. Experimental Results

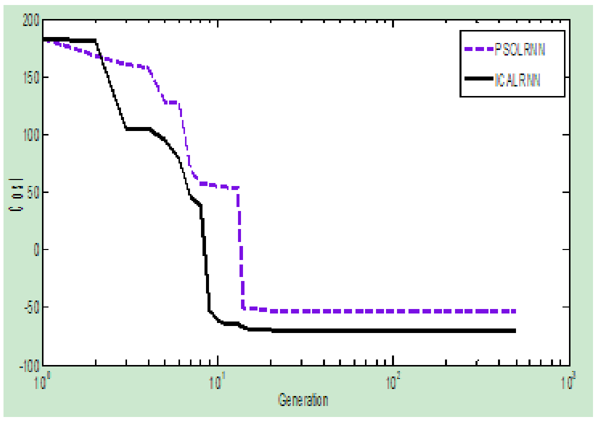


Fig. 8.The average of values for 100 instances ofSAT150

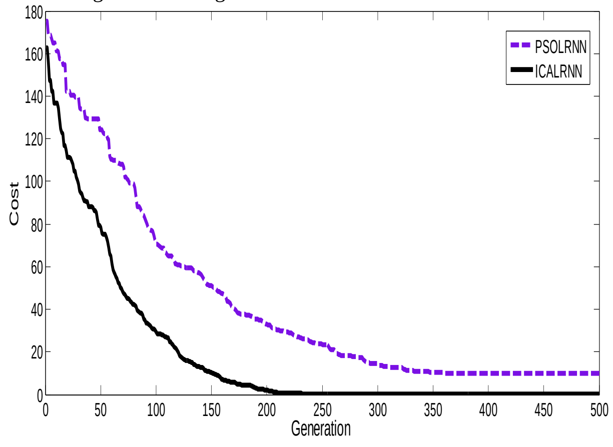


Fig. 9.The average of values for 100 instances ofSAT300

In this paper, first a new algorithm that combines ICA and LR to solve SAT problem has been proposed. The results obtained after solving some instances of the SAT showed that ICALR is computationally efficient in solving these problems. ICALR was faster than PSOLR.

In PSOLR, we use PSO algorithm for solving SAT problem. ICALR provides solutions that are comparable to these approaches. In terms of the solution quality, the ICALR provided a “best solution” with a lower cost than PSOLR for problem sizes larger than 200 variables.

In Table III, the parameters for ICA algorithm and PSO algorithm are shown.In some of instances ICALR may trap into local optima that it shows in Figs. 8, 9 and Table IV. The results are in 1000 iteration for algorithms.

TABLEIII: THEPARAMETERSFORIMPLEMENTATION

Algorithm	Parameters
ICA	Num. Of Countries = 80, Num. Of Initial Imperialists = 8, Num. Of Decades = 1000, Revolution Rate = 0.3, Assimilation Coefficient= 2, Zeta = 0.02.
PSO	C1 = 1.5, C2 = 1.5, $\beta = {iter}/{iter_{max}}$,S=-0.5, Particle Size = 80, Max Iter. = 1000

TABLE IV: COMPARING THE BEST COST FOR PSOLR AND ICALR

Name	PSOLA	ICALR (proposed algorithm)
Uf50-218/uuf50-218	-120.06	**-130.76**
Uf75-325/uuf75-325	-94.01	**-99.93**
Uf100-430/uuf30-430	-90.00	**-93.17**
Uf125-538/uuf125-538	-85.42	**-89.50**
Uf150-645/uuf150-645	-76.20	**-77.44**
Uf175-753/uuf175-753	-60.00	**-72.55**
Uf200-860/uuf200-860	-57.77	**-66.34**
Uf225-960/uuf225-960	-52.51	**-63.7**
Uf250-1065/uuf250-1065	-50.11	**-59.00**

Secondly, a new hybrid Hopfield neural network-Imperialist competitive algorithm (HNNICA) for solving SAT problem has been presented. The algorithm consists of a Hopfield neural network which manages the problem’s constraints, hybridized with ICA which improves the solution obtained from the network. Simulations in a set of SAT benchmark problems have shown very good performance of the algorithm. The results are shown in Tables V, VI.

TABLE V: COMPARING THE BEST COST OF ALGORITHMS

Instance/Cost	HRNN	ICALR (proposed algorithm)	HNNICA (proposed algorithm)
SAT 50	-45	-130	**-135**
SAT 100	-30	-93	**-123**
SAT 125	-26	-89	**-116**
SAT 150	-20	-77	**-109**
SAT 200	-18	-66	**-97**
SAT 250	-10	-63	**-91**
SAT 300	-8	-36	**-80**

TABLE VI. COMPARING THE RUN TIME OF ALGORITHMS

Instance/Time (second)	HRNN	ICALR (proposed algorithm)	HNNICA (proposed algorithm)
SAT 50	10.50	20.45	**18.12**
SAT 100	37.21	45.70	**36.91**
SAT 125	35.64	43.55	**30.34**
SAT 150	82.23	82.61	**73,62**
SAT 200	163.01	150.12	**89.00**
SAT 250	257.51	264.65	**120.55**
SAT 300	321.87	289.22	**158.22**

HNNICA algorithm, enhance the global search capability of the HNN algorithm. This idea balances the exploration and exploitation abilities of the proposed algorithm, using ICA and HNN. We examined the proposed algorithm in several SAT problems. From the results of experimentations on SAT problems, we observed that the proposed ICALR and HNNICA algorithms, especially HNNICA is superior to HNN.

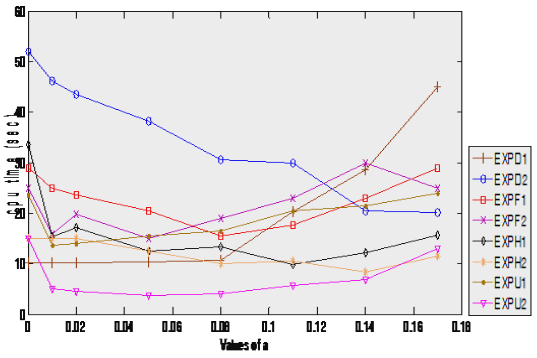


Fig. 10. CPU time versus α in LPPH algorithm

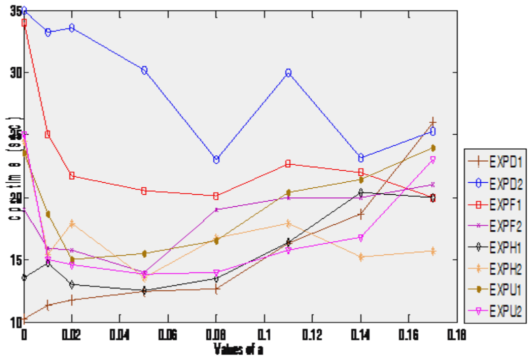


Fig. 11. CPU time versus α in HNNICA algorithm

Figs. 10, 11 show the results for eight SAT problems. These problems are solved by the LPPH. In the LPPH, α parameter is attenuation coefficient of weights. The performance of LPPH for solving the SAT strongly depends the value of α.The good value for α depends the problem. In Fig. 11, show CPU time using changing the value of α. Fig. 11 shows the results of HNNLR for selecting good value for α is somewhat relaxed than the LPPH.

VI. CONCLUSION AND FUTURE WORK

In this paper is proposed two algorithm for Solving SAT problems. First, a new algorithm that combines ICA and LR to solve SAT problem has been proposed. Secondly, this paper presents a hybrid Hopfield network (HNN)-ICA to solve the SAT problem. In this proposed algorithm, Hopfield neural network (HNN) manages the problem's constraints and ICA algorithm searches for high quality solutions and minimum cost. HNN hybridized with ICA which improves the solution obtained from the network. Simulations in a set of SAT benchmark problems have shown very good performance of the algorithm. HNNICA algorithm, enhance the global search capability of the HNN algorithm. This idea balances the exploration and exploitation abilities of the proposed algorithm, using ICA and HNN. Simulations are observed that the ICALR and HNNICA algorithms have a good performances, especially HNNICA is superior to HNN. In the future, we will work on the effect of the PSO algorithm on the performance of the ICA algorithm. The results show the HNNLR for selecting good value for α is somewhat relaxed than the LPPH.

ACKNOWLEDGEMENTS

We would like to thank the creator of the ICA algorithm Atashpaz-Gargari for his help of answering our emails while understanding and tuning the parameters of algorithm. In addition, we would like to thank Mr. JavadAlikhanikoupaei forhis help while doing this research.We are grateful for the reviewers for their helpful commentsand suggestions to improve our work.

REFERENCES

[1] E. A. Gargari and C. Lucas, "Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition,"*IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 4661-4667, 2007.

[2] H. Bahrami, K. Faez, and M. Abdechiri, "Imperialist Competitive Algorithm using Chaos Theory for Optimization," *UKSim-AMSS 12th International Conference on Computer Modelling and Simulation Cambridge UK*, pp. 98-103, 2010.

[3] M. Abdechiri, K. Faez, and H. Bahrami, "Neural Network Learning based on Chaotic Imperialist Competitive Algorithm," *The 2nd International Workshop on Intelligent System and Applications (ISA2010)*, 2010.

[4] J. Hopfield, "Neurons and Physical Systems with Emergent Collective Computational Abilities," in *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–8, 1982.

[5] J. Balicki, A. Stateczny, and B. Zak, "Genetic Algorithms and Hopfield Neural Networks to Solve Combinatorial Optimization Problems,"*Applied Mathematics and Computer Science*, vol. 10, no. 3, pp. 568–92, 1997.

[6] S. S. Sanz and J. P. Figueras, "Optimal Solution to Crossbar Packet-Switch Problems using a Sequential Hopfield Neural Network," *Neurocomputing*, vol. 70, pp. 2603–2607, 2007.

[7] M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem-Proving," *Communications of the ACM*, vol. 5, pp. 394-397, 1962.

[8] D. Loveland,"Automated Theorem Proving: A Logical Basis,"*North-*Holland, 1978.

[9] M. Bertran, S. Lakhdar, and G. Eric, "Tabu Search for SAT,"*In Proc. of the 14th National Conference on Artificial Intelligence and 9th* Innovative *application of Artificial Intelligence Conference*, pp. 281-285, 1997.

[10] W. M. Spears, "Simulated Annealing for Hard SatisfiabilityProblems," In second *DIMACS implementation challenge: cliques*, vol. 26, pp.533-558, 1996.

[11] A. Kenneth, P. Andre, and M. Spears, "Using Genetic Algorithm to Solve NP-Complete Problems," *In Proc. of the Third Int. Conf. on Genetic Algorithms*, pp. 124-132, 1989.

[12] J. K. Hao and R. Dorne, "A New Population-based Method for SatisfiabilityProblems," In *Proc. of the 1th European conf. on Artificial Intelligence*; pp. 135-139, 1994.

[13] C. Fleurent and G. A. Ferland, "Genetic and Hybrid Algorithms for Graph Coloring," *Annals of Operations Research*, vol. 3, pp. 437-461, 1997.

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing, "*Science*, vol. 4598, pp. 671–680, 1983.

[15] B. Selman, H. Kautz, and B. Cohen, "Noise Strategies for Improving Local Search. *In 22th Nat. Conf. on Artificial Intelligence California*, pp. 337–343, 1994.

[16] B. Selman, H. Levesque, and D. Mitchell, "A New Method for Solving Hard Satisfiability Problems," *Proceedings of the AAAI Conference San Jose*, pp. 440–446, 1992.

[17] M. Bertrand, S. Lakhdar, and G. Eric, "TabuSearch for SAT," *In Proc. of the AAAI-97/IAAI-97*, pp. 281–285, 1997.

[18] M. Nagamatu and T. Yanaru, "Lagrange Programming Neural Networks with Polarized High-Order Connections for Satisfiability Problems of Propositional Calculus," *The 3rd International Conferenceon Fuzzy Logic, NeuraI Nets and Soft Computing*, pp. 233-235, 1994.

[19] M. Nagamatu and M. Yanaru, "On the Stability of Lagrange Programming Neural Networks for Satisfiability Problems of Propositional Calculus," *Neurocomputing*, vol. 13, no. 2-4, pp. 440-6, 1992.