2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

# A Guided Hopfield Evolutionary Algorithm with Local Search for Maximum Clique Problem

Gang Yang*†, Xirong Li*‡, Jieping Xu*†, Qin Jin*†, Hui Sun*†,
*Key Lab of Data Engineering and Knowledge Engineering, MOE, Renmin University of China, Beijing, China
†School of Information, Renmin University of China, Beijing, China

***Abstract*—In this paper, a novel hybrid evolutionary algorithm combining a *Hopfield* net and a local search strategy is proposed to solve maximum clique problem. The algorithm makes full use of powerful searching capability of *Hopfield* net and probabilistic statistic feature of estimation of distribution algorithm to produce wider search in global solution domain. In particular, a possible extension way correlated with local search optimization is introduced to affect the mutation probability thus to produce guided evolution. Experiments on the popular DIMACS benchmark demonstrate that the hybrid evolutionary algorithm produces comparable and better results than other compared algorithms, including *EA/G* which is a state-of-the-art algorithm in the field of evolutionary computation.**

## I. Introduction

Maximum clique problem (MCP) is a paradigmatic combinatorial optimization problem with many relevant applications, such as computer vision and pattern recognition. MCP is NP-hard, and numerous approximation algorithms are proposed to solve MCP, including neural networks, evolutionary algorithms and swarm intelligent algorithms.

To solve MCP, EA/G, the state-of-the-art algorithm, is an evolutionary algorithm obtaining guided mutation to generate offspring through combination of global statistical information and the location information of solutions found so far[3]. However, EA/G obtaining swarm intelligence needs massive calculations in practical applications. Subsequently, based on EA/G and reactive search optimization principles[7], R-EVO algorithm regarded as an efficient hybrid algorithm was proposed[2]. However, the capability of R-EVO naturally relies on the search ability of reactive local search (RLS), even depends on large memory store to form complex individuals. Additionally, swarm intelligence algorithms are able to search wide solution spaces by increasing the individual bifurcations. Specially, in swarm intelligence algorithms, ant colony optimization algorithm (ACO) obtains more competitive results on solving MCP[6][10], but it has weak stability when solving MCP for its swarm characteristics.

Neural networks were also wildly applied to solve combinatorial optimization problems. Previous studies using neural networks for MCP are mainly about *Hopfield*-type algorithms[5][8], which express powerful solving ability. However, *Hopfield*-type algorithms possess serious local minima problem. To overcome the problem, feasible initial positions will provide large possibility to search an efficient domain obtaining optimal solutions. Therefore, algorithm with multiple starts on different initial positions is a basic optimization way to improve *Hopfield-type* net.

A more powerful way to solve the local minima problem is through combining *Hopfield-type* algorithms with evolutionary algorithms to create more feasible initial positions. A discrete competitive *Hopfield-type* algorithm proposed by Wang et al. for maximum diversity problems exactly adopts this way to increase the probability of finding a feasible initial position to enhance performance[5]. However, these algorithms do not consider their intermediate states of algorithm processing, which are useful information to search final solutions.

Inspired by EA/G and wang's work, we assess whether the hybrid of *Hopfield* net with local search and estimation of distribution algorithm (EDA) [9] could produce an efficient algorithm. Moreover, we want to confirm whether local search strategy related with MCP could help continuous *Hopfield* net to extend its searching areas. Therefore, we present a local search strategy about repairing unfeasible results and solution possible extension to fit for the continuous *Hopfield* net and the probability model of EDA. In this paper, we focus on combining the continuous *Hopfield* net with EDA evolutionary algorithm to expand the searching spaces, and propose a guided Hopfield evolutionary algorithm (GHEA) with local search for MCP. The main features of GHEA are as follows:

- Efficient hybrid: Hopfield net does not contain sufficient ability to search optimal solutions, but the hybrid of EDA and local search strategy greatly optimize its performance.
- Suitable probability model: New initial positions are produced by guided mutation and possible extension corresponding to concrete MCP instances. Useful local information guides the probability model of initial value setting.

The rest of this paper is organized as follows. Section II demonstrates GHEA in detail. Experimental study and comparisons are shown in Section III. Section IV gives the conclusion of the paper.

## II. Guided Hopfield Evolutionary Algorithm with Local Search

In this section, we propose a hybrid algorithm which combines a continuous Hopfield net, an EDA and a local search strategy effectively to directly solve maximum clique problem.

‡ Corresponding author. E-mail address: xirong@ruc.edu.cn. This research was partially supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (No.14XNLQ01).

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

MCP can be defined as: let $G = (V, E)$ be an undirected graph, $V = \{1, 2, ..., n\}$ its vertex set, $E \subseteq V \times V$ its edge set, and $G(S) = (S, E \cap S \times S)$ the subgraph induced by $S$, where $S$ is a subset of $V$. A graph $G = (V, E)$ is complete if all its vertices are pairwise adjacent, i.e., $\forall i, j \in V$ and $(i, j) \in E$. A clique $K$ is a subset of $V$ such that $G(K)$ is complete. The MCP asks for a clique of maximum cardinality. Our proposed algorithm, named *GHEA*, is represented in Algorithm 1.

**Algorithm 1** *GHEA with local search for MCP*

1: $k = 0$;
2: initialize $U$ and $V$ (randomly in [0 1]);
3: initialize $Q_{best}$, $S_{best}$ and probability model $P$: $Q_{best} = \emptyset$, $S_{best} = 0$, $P = 0.5$;
4: **repeat**
5: execute the continuous *Hopfield* net for MCP;
6: get the convergent solution $V$ of *Hopfield*;
7: $C_s$ = $repair(d, V)$;
8: update $p_v$ by searching the possible extension of $C_s$;
9: **if** $|C_s| > (S_{best} - \Delta)$ **then**
10: $Q_{best}$ = $C_s$;
11: $S_{best}$ = $|C_s|$;
12: **end if**
13: update probability model: $p = (1 - \lambda)p + \lambda p_v$;
14: **if** (rand()$> \mu$) **then**
15: reset $p$ to $1 - p$ ;
16: **end if**
17: generate a new starting point $V$ for *Hopfield* net with $mutate(Q_{best}, p, \beta)$;
18: $U = 0$;
19: $k = k + 1$;
20: **until** ($k > k_{max}$ outer epoch or stability situation are satisfied)

In the proposed algorithm, Hopfield net is the core to solve MCP because of it obtaining deterministic search in a specified domain. Given a deterministic input, *Hopfield* net searches a broad solution domain utilizing a gradient descent method. Specifically, the input affects the searching capability, so an effective way to improve *Hopfield* net is the bifurcation of input values. To solve MCP, the connection relationship in Hopfield net is defined as:

$$d_{ij} = \begin{cases} 1 & if (i, j) \in E \\ 0 & otherwise \end{cases} \quad (1)$$

The state of a neuron $v_i$ is determined by

$$v_i = \begin{cases} 1 & if \quad the \ \#\text{i} \ vertex \ is \ in \ clique \\ 0 & otherwise \end{cases} \quad (2)$$

The energy function, momentum function and activation function are shown as follows:

$$E = A \sum_i v_i + \frac{B}{2} \sum \sum (1 - d_{ij}) v_i v_j \quad (3)$$

$$u_i(t + 1) = k u_i(t) + \alpha( \sum_{j=1, j \neq i}^{n} w_{ij} v_j(t) - I) \quad (4)$$

$$v_i = \frac{1}{1 + e^{-u_i/\varepsilon}} \quad (5)$$

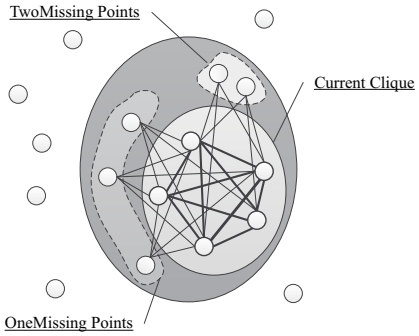


Fig. 1. Possible extension of current clique in GHEA. The figure illustrates a new point set defined as "TwoMissing Points" to enhance the searching domain in our algorithm.

According to the above functions, the weights $w_{ij}$ and the bias oscillation $I$ are defined as: $w_{ij}$=$-2B(1 - d_{ij})$, $w_{ii}$=0 and $I$=$A$. In the original continuous *Hopfield* net[12], neurons' outputs are restricted as: if $v_i > \delta$, then $v_i$=1; otherwise, $v_i$=0. Normally, $\delta$ is set 0.9. In step 5 and 6 of GHEA, *Hopfield* net searches solutions during $L$ iteration limitation. It has been validated that *Hopfield* can get convergence definitely when weights $W$ satisfied previous states. In *GHEA*, the iteration limitation $L$ is fixed to 500 for increasing process speed. Subsequently, if the energy maintains a stable value for a dozens of iterations, *Hopfield* net is also terminated to export the convergent solutions.

The *repair* function is designed to adapt the characteristics of the continuous *Hopfield* network. In a continuous *Hopfield* network, neurons saturate to real numbers between 0 and 1. Moreover, the neuron outputs directly reflect accessed probability about the problem point during the searching process. Thus, the repair process of *GHEA* uses a deterministic way to extract and extend the solution of *Hopfield* net. To produce a deterministic clique, the repair function is defined:

**function** $repair(d, V)$

1: find point index $I_v$ of current solution $V$ with $v_i$ =1;
2: /*Extraction*/
3: **for** ($i$=1 to $size(I_v)$) **do**
4: **if** (all $d_{ij} = 1$ where $i, j \in I_v$ but $i = j$) **then**
5: break;
6: **else**
7: delete $i$ from $I_v$ where point $i$ has the least degree in $I_v$;
8: **end if**
9: **end for**
10: /*Extension*/
11: **for** ($i$=1 to $size(I_v)$) **do**
12: **if** no existed points connect to all points of $I_v$ **then**
13: break;
14: **else**
15: add a point who connects to all points of $I_v$;
16: **end if**
17: **end for**

Two sets are involved in the mutation probability decision: One is *oneMissing* which contains the points connected to all

but one element of the clique, and the other is *twoMissing* containing the points connected to all but two elements of the clique, shown in Fig.1. The points in the set of *oneMissing* can build a valid clique whose size is equal to current clique. The *twoMissing* points help change the direction of rebuilding the current clique to access a wider range of solutions. In the possible extension process, based on *oneMissing* and *twoMissing*, an overall neuron probability $P_v$ is gotten to reflect the characteristic of local search. In our algorithm, $P_v$ decides the probability modification of estimation of distribution. Followed by *repair* function, probability model $P_v$ is summarized by differentiating the missing points of current clique. $P_v$ is classified to three levels probability. $p_{vi}$=1 where point $i$ belongs to *oneMissing* set; $p_{vi}$=0.5 where node $i$ is one of points in *twoMissing* set; otherwise, $P_{vi}$=0;

Subsequently, *GHEA* judges the current clique whether is a new feasible clique. If the current clique is within the range of feasible solutions, the best clique $Q_{best}$ and its size $S_{best}$ will be updated to transform its searching domain. Parameter $\Delta$ controls the tolerant degree of acceptable solution. $\Delta = 1$, normally deployed, means that $Q_{best}$ perturbs in domains of the same size clique, which indirectly reflects the affection of *oneMissing* set. Similarly, *twoMissing* set is corresponding of $\Delta = 2$. Parameter $\Delta$ greatly influences the searching process because it may modify the searching direction absolutely.

For adapting a serial execution environment, the probability vector $p$ is updated in the same way as in the Population-Based Incremental Learning algorithm [11] :

$$p_i = (1 - \lambda)p_i + \lambda p_{vi} \tag{6}$$

for $i = 1, 2, ..., n$, and $\lambda \in (0, 1]$ is the learning rate. The probability vector $p$ not only indicates the feature of local situation, but also reflects the solution distribution of global searching probability. Moreover, $p_{vi}$ decreases the computing complexity.

In step 17 of algorithm 1, *GHEA* perturbs the current solution using EDA *mutation* function as follows:

**function** $mutate(v, p, \beta)$
1: **for** ($i$=1 to $|v|$) **do**
2: **if** rand()< $\beta$ **then**
3: **if** rand()< $p_i$ **then**
4: set $v_i(t+1) = 1$;
5: **else**
6: set $v_i(t+1) = 0$;
7: **end if**
8: **else**
9: $v_i(t+1) = v_i(t)$
10: **end if**
11: **end for**
12: return $v$

where rand() produces a random number distributed uniformly in [0,1]. In the EDA mutation, a bit is sampled from the probability vector $p$ randomly or directly copied from the current best clique $v$, which is controlled or balanced by the parameter $\beta$. The larger the $\beta$, the more elements of the new starting point are sampled from the probability vector $p$. Two

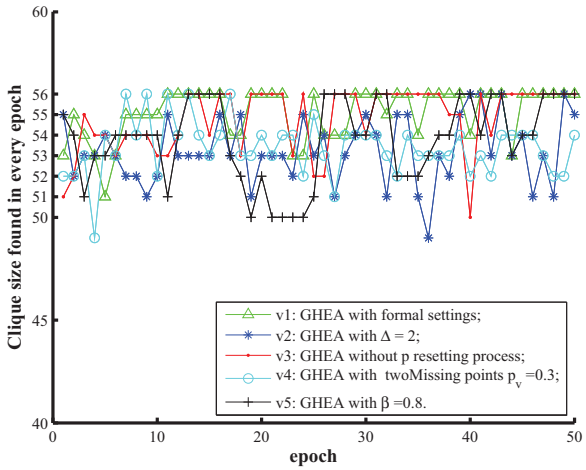


Fig. 2. Different running states of *GHEA* with different settings. The experiments are tested on C500.9 of MCP. The average clique sizes gotten by five *GHEA* versions are sequentially 55.1, 53.26, 44.82, 53.42 and 53.82. This figure illustrates the different parameter affections. Moreover, it shows our algorithm can escape from previous local minima in epoch process.

random factors can enhance the diversity of searching situations. The new starting point has a large probability to position as a feasible initial value to set *Hopfield* net. Subsequently, the inner states $U$ of *Hopfield* net will be interferences because its outer searching states already fall in a new domain. Thus, $U$ is reset to zero to banish the interference.

Through searching by *Hopfield* net and executing guided evolution by EDA and local search, *GHEA* builds a gradually improving process with new better starting situations in every epoch. The calculation of *Hopfield* net takes time $O(n^2)$, the computing complexity of probability model updating is $O(n)$, and the process of repairing and possible extension take time $O(n)$. $n$ is the number of neurons. Therefore, the whole complexity of *GHEA* is $O(n^2)$ due to the main effect of *Hopfield* net.

## III. Experiments

In this section, we study *GHEA* experimentally and compare it with EA/G, the state-of-the-art algorithm in evolutionary field, on the DIMACS benchmark problems. We also compare the performance of *GHEA* with ant colony optimization (ACO) algorithm and multi-start Hopfield algorithm on these test problems. Our experiments were compiled by Matlab and were executed on an Intel 2.5GHz machine with 4GB RAM. In order to make comparison impartially, hardware differences are taken into account. We realized EA/G algorithm by ourselves, and summarized its execution time in our experimental environment. The clique sizes found by EA/G are adopted as same as the values in [3].

To discuss effect of different parameters and settings in *GHEA*, we compared the results obtained by differential instances of *GHEA*. On solving MCP C500.9 instance, we built different test versions of *GHEA* through modifying individual parameter in every running time. Fig. 2 shows the comparison results on the clique size gotten by different instances of

TABLE I. COMPARISON AMONG MULTI-START HOPFIELD, EA/G, ACO, AND *GHEA* FOR SPECIFIC MCP INSTANCES.

| Instance | DIMACS | Multi-Start Hopfield | | | EA/G[3] | | | ACO[10] | | | GHEA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg(std) | Best | time(s) | Avg(std) | Best | time(s) | Avg(std) | Best | time(s) | Avg(std) | Best | time(s) |
| c125.9 | 34 | 30.7(1.9) | 34 | 1.79 | 34.0(0.0) | 34 | 1.5 | 33.2(0.5) | 34 | 1.79 | 34.0(0.0) | 34 | 1.56 |
| c250.9 | 44 | 36.8(0.8) | 38 | 2.98 | 44.0(0.0) | 44 | 2.9 | 42.3(0.9) | 44 | 3.36 | 44.0(0.0) | 44 | 2.97 |
| c500.9 | 57 | 50.2(1.3) | 54 | 9.71 | 55.2(0.9) | 56 | 5.7 | 55.2(1.8) | 56 | 8.77 | 56.1(0.3) | **57** | 8.68 |
| c1000.9 | 68 | 53.1(3.5) | 61 | 29.16 | 64.4(1.4) | 67 | 21.2 | 61.4(3.9) | 65 | 29.67 | 64.9(1.2) | 67 | 25.82 |
| MANN_a27 | 126 | 122.4(2.4) | 126 | 16.07 | 126.0(0.0) | 126 | 12.1 | 125.4(0.8) | 126 | 15.63 | 126.0(0.0) | 126 | 16.07 |
| MANN_a45 | 345 | 321.9(4.9) | 339 | 106.51 | 343.7(0.7) | 345 | 80.2 | 340.4(1.2) | 342 | 109.13 | **343.8(0.4)** | 345 | 111.35 |
| brock200_2 | 12 | 9.1(1.0) | 11 | 2.13 | 12.0(0.0) | 12 | 1.8 | 9.4(1.1) | 12 | 2.23 | 12.0(0.0) | 12 | 2.12 |
| brock200_4 | 17 | 11.3(1.1) | 14 | 2.47 | 16.5(0.5) | 17 | 2 | 14.8(0.7) | 16 | 2.25 | **16.8(0.4)** | 17 | 2.38 |
| brock400_2 | 29 | 21.4(1.4) | 23 | 4.43 | 24.7(0.4) | 25 | 3.7 | 23.8(0.6) | 25 | 4.11 | 24.7(0.3) | 25 | 4.16 |
| brock400_4 | 33 | 20.4(1.9) | 25 | 4.59 | 25.1(2.6) | 33 | 3.9 | 24.9(3.5) | 33 | 5.36 | **27.9(2.1)** | 33 | 4.51 |
| brock800_2 | 21 | 18.1(0.8) | 21 | 9.68 | 20.1(0.4) | 21 | 8.9 | 19.5(0.6) | 21 | 11.86 | **20.4(0.3)** | 21 | 10.14 |
| hamming8-4 | 16 | 16.0(0.0) | 16 | 2.3 | 16(0.0) | 16 | 2 | 16(0.0) | 16 | 2.38 | 16.0(0.0) | 16 | 2.38 |
| hamming10-4 | 40 | 35.7(1.3) | 40 | 20.74 | 39.8(0.6) | 40 | 16.7 | 39.1(0.2) | 40 | 20.49 | 39.5(0.6) | 40 | 20.32 |
| keller4 | 11 | 7.8(0.9) | 9 | 1.73 | 11.0(0.0) | 11 | 1.5 | 11(0.0) | 11 | 1.77 | 11.0(0.0) | 11 | 1.73 |
| keller5 | 27 | 20.6(0.9) | 24 | 14.28 | 26.9(0.3) | 27 | 10.7 | 25.9(1.3) | 27 | 13.01 | 26.4(0.6) | 27 | 13.26 |
| keller6 | 59 | 48.2(2.3) | 53 | 84.41 | 53.4(1.2) | 56 | 63.1 | 51.2(2.1) | 53 | 68.09 | 53.2(1.5) | 56 | 80.16 |
| p_hat300-1 | 8 | 7.1(0.2) | 8 | 2.97 | 8.0(0.0) | 8 | 2.4 | 8.0(0.0) | 8 | 2.81 | 8.0(0.0) | 8 | 2.81 |
| p_hat300-2 | 25 | 23.9(0.4) | 25 | 2.81 | 25.0(0.0) | 25 | 2.3 | 24.9(0.1) | 25 | 2.71 | 25.0(0.0) | 25 | 2.62 |
| p_hat300-3 | 36 | 32(0.5) | 34 | 3.41 | 36.0(0.0) | 36 | 2.7 | 34.6(0.9) | 36 | 3.6 | 36.0(0.0) | 36 | 3.26 |
| p_hat700-1 | 11 | 9.0(0.0) | 9 | 8.67 | 11.0(0.0) | 11 | 6.6 | 9.1(1.6) | 11 | 7.18 | 11.0(0.0) | 11 | 8.25 |
| p_hat700-2 | 44 | 38.4(0.8) | 40 | 11.39 | 44.0(0.0) | 44 | 8.9 | 43.4(0.6) | 44 | 11.14 | 43.9(0.2) | 44 | 10.62 |

*GHEA*. Fig. 2 indicates that *GHEA* is not seriously sensitive to those parameters. Every *GHEA* instance can find good solutions, and the only difference is the consuming time. In detail, from Fig. 2, it is illustrated that *GHEA* with formal settings has the best execution on the aspect of average clique size in one running process. Although *GHEA v3* can get better cliques earlier, its stability is too poor to search more valuable solutions in the following epochs. Other algorithms instances also have their visible drawbacks in the process. Therefore, in our simulations the parameters in *GHEA v1* were adopted. In the simulation, the parameters of *GHEA* with formal settings are set as follows: $\Delta$=1, $p_v$=0.6, $\beta$=0.5, $\mu$=0.5 .

In simulations, different type of MCP instances are tested to reveal *GHEA* performance. The set of algorithm column reports the average maximum clique found(with the corresponding standard error), the overall maximum and the average execution time for 30 runs or the reference time of original literature. Some data with bold face mean the best results found in our experiments. The results show that *GHEA* has a significant superiority to EG/A in the aspect of average size and standard error. In the case of dense random graph C500.9, *GHEA* ever found a better clique with 57 nodes. And for large instances C1000.9 and MANN_a45, the standard error is about 14% and 45% better than EG/A. For the small instances, EG/A, *GHEA* and ACO can always locate the maximum clique. The multi-start Hopfield deployed just with random initial positions gained the worst results among four algorithms. It illustrates the probability model and local search strategy guided the evolutionary procedure to provide appropriate initial inputs for Hopfield net in *GHEA*. The probability model of EDA produces prohibition effect to restrain *GHEA* from searching the areas already searched. Local search strategies can help more effectively determine the position of a new starting point for *Hopfield* net in *GHEA*. The multi-start Hopfield lacks the two parts, therefore its performance is worse. In a word, our algorithm has superior performance in the experiments and shows the guided searching capability of local search strategy.

## IV. CONCLUSIONS

In this paper, we have presented a hybrid algorithm, named *GHEA*, based on *Hopfield* net combined with EDA and local search for the maximum clique problem. According to the characteristic of continuous *Hopfield* net, the algorithm utilizes a specific repair process to efficiently enhance the quality of Hopfield intermediate state. Simultaneously, the perturbation based on the idea of EDA probability can generate a new initial situation for *Hopfield* to guide for heuristic searching. Our algorithm can escape from local minima efficiently and search wider domains. We have tested our algorithm on a set of MCP instances of different difficulties, outperforming the state-of-the-art approach to some extent on MCP.

## REFERENCES


[1] Szab S. Parallel algorithms for finding cliques in a graph. Journal of Physics: Conference Series. IOP Publishing, 2011, 268(1): 012030.

[2] Brunato M, Battiti R. R-EVO: a reactive evolutionary algorithm for the maximum clique problem. Evolutionary Computation, IEEE Transactions on, 2011, 15(6): 770-782.

[3] Zhang Q, Sun J, Tsang E. An evolutionary algorithm with guided mutation for the maximum clique problem. Evolutionary Computation, IEEE Transactions on, 2005, 9(2): 192-200.

[4] Pehlivanoglu Y V. A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks. Evolutionary Computation, IEEE Transactions on, 2013, 17(3): 436-452.

[5] Wang J, Zhou Y, Yin J, et al. Competitive Hopfield network combined with estimation of distribution for maximum diversity problems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2009, 39(4): 1048-1066.

[6] Fenet S, Solnon C. Searching for maximum cliques with ant colony optimization. Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2003: 236-245.

[7] Battiti R, Protasi M. Reactive local search for the maximum clique problem 1. Algorithmica, 2001, 29(4): 610-637.

[8] Gang Yang, Nan Yang, Junyan Yi, and Zheng Tang. An improved competitive Hopfield network with inhibitive competitive activation mechanism for maximum clique problem. Neurocomputing, 2014, 130(23) : 28-35.

[9] P. Larranaga, Estimation of distribution algorithms a new tool for evolutionary computation. Kluwer Academic Publishers, 2001.

[10] Solnon C, Fenet S. A study of ACO capabilities for solving the maximum clique problem. Journal of Heuristics, 2006, 12(3): 155-180.

[11] Baluja S. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

[12] Hopfield J J, Tank D W. Neural computation of decisions in optimization problems. Biological cybernetics, 1985, 52(3): 141-152.