A Bi-level Blocked Estimation of Distribution Algorithm with Local Search for Maximum Clique Problems

Yan Zhang , Qun Dang, Zhu Jiang, YongXuan Huang

Abstract —**Maximum Clique Problem (MCP) is a complicated deceptive problem for estimation of distribution algorithms (EDAs). The univariate EDAs cannot utilize the correlations of the variables and the advanced EDAs perform poor due to the expensive computational cost in building the appropriate probability models. In this paper, by utilizing the special structure of MCP, a new Bi-level Blocked Probability model (BBP) is constructed, which achieves the relationships utilizing in a bivariate probability model at the computational cost of univariate probability model. Integrating promising neighborhood search techniques, a new EDA algorithm, called Bi-level Blocked Estimation of Distribution Algorithm (BBEDA) is proposed for MCP. Comparative experiments on extensive DIMACS Benchmark instances show that the proposed BBEDA can be competitive with the evolutionary algorithm with guided mutation (the best evolutionary algorithm reported so far) in terms of solution quality and computational performance.**

I. Introduction

Maximum Clique Problem (MCP) is to find a vertex subset in a given graph, which is pair wise adjacent in graph and has the maximum cardinality. It is one of the problems that have been proven to be NPC and there is no polynomial time algorithm for approximating the maximum clique within a factor of $n^{1-\varepsilon}$ unless P=NP [1], where n is the number of vertices of the graph. MCP has wide applications in practice and much effort has been devoted to it. Traditional methods designed to find the optimal solution cannot scale to the large size instances due to the NPC characteristics of MCP. Therefore, many researchers have focused on developing efficient heuristics for MCP. Genetic Algorithm, Simulated Annealing, Ant Colony Systems and their hybrid algorithms have been applied to MCP [2]. But the MCP had been found to be a complicated deceptive problem for EAs and the general conclusion was that EAs "need to be heavily customized in order to be competitive with traditional approaches, and that they are computationally very expensive" [3].

Yan Zhang is with Dept. of Electronic & Information Engr, Xi'an JiaoTong University, Xi'an, Shaanxi, 710049, China (e-mail: zhangyan.nowed@gmail.com).

Qun Dang is with Dept. of Electronic & Information Engr, Xi'an JiaoTong University, Xi'an, Shaanxi, 710049, China (e-mail: dqmama@163.com).

Zhu Jiang is with Dept. of Electronic & Information Engr, Xi'an JiaoTong University, Xi'an, Shaanxi, 710049, China (e-mail: hill5525 @163.com)

YongXuan Huang is with SYSTEM ENGINEERING Lab., Xi'an JiaoTong University, Xi'an, Shaanxi, 710049, China (e-mail: yxhuang@sei.xjtu.edu.cn).

Recently, Qingfu Zhang proposed an evolutionary algorithm with guided mutation (EA/G) for MCP [4]. EA/G works with a population of solutions and generates new solutions by *guided mutation*, which combines the conventional mutation operator with the EDA offspring generating scheme. Though due to the inherent computational overheads in constructing the probability model, EA/G performs worse than the simple stochastic local search (SLS) algorithms [5], such as Dynamic Local Search Algorithms (DLS_MC) [6] or Reactive Local Search Algorithms (RLS) [7], EA/G had been the best EAs algorithms for MCP so far in terms of solution quality and computational overheads.

EA/G built an order-1 probability model of distributions, which had not utilized the dependence relationships among variables. But it did outperform the order-2 probability model, such as MIMIC probability model, on MCP instances, which can investigate pairwise inter-vertex correlations. The reason is not that the correlations among the variables are unimportant, but that the distributions of locally optimal solutions are much more complicated than the MIMIC could approximate.

In this paper, a new bi-level blocked estimation of distribution algorithm (BBEDA) with local search is proposed for MCP. By exploiting the special inherent structure of MCP, a new relational probability model, bi-level blocked probability model (BBP) was built to represent the distributions over the good solutions in MCP.

With domain knowledge, the proposed BBP model partitioned the vertex set into several groups in such a way that no more than one vertex could appear in a clique in each group. The structure of the search space is reorganized and there exist mutually exclusive relations among the vertices within each group and contest relations among groups. The BBP model then focus on developing these relationships among vertices.

The BBP model has a two-level structure: bottom level and top one. In the bottom level, based on the inherent mutually exclusive relations, a pseudo-Boltzman distribution probability model is constructed to estimate the probability of each vertex appearing in the optimal solution; In the top level, an ordinal estimation model is built to determine the priority of the group.

The new probabilistic modeling decreases the representational complexity of distribution by partitioning the estimation of the whole search space into several sub-estimations of the sub-search spaces, and it contains the relationships identification among vertices at a low

978-1-4244-1823-7/08/$25.00©2008 IEEE

computational cost of univariate EDAs.

In order to avoid rapid convergence, a diversification technique is introduced to the new bilevel blocked probability model. By assigning a high probability to the unvisited vertices, the new probability model causes a bias towards the new and promising areas in the search space.

The rest of this paper is organized as follows: section II introduces bi-level blocked estimation of distribution model after analyzing the special structure in the MCP. BBEDA for the MCP is presented in section III. Experimental study and comparisons are given in section IV. Conclusions are presented in section V.

II. Bi-level Blocked Probability Model for MCP

Maximum Clique Problem (MCP) is a complicated deceptive problem for estimation of distribution algorithms (EDAs). The univariate EDAs cannot utilize the correlations of the variables and the advanced EDAs perform poor due to the expensive computational cost in building the appropriate probability models. In this paper, by utilizing the special structure of MCP, a new Bi-level Blocked Probability model (BBP) is constructed, which achieves the relationships utilizing in a bivariate probability model at the computational cost of univariate probability model. Before giving the BBEDA model, we first introduce the notations and definitions used in this paper.

A. Notations and Definitions

Let $G(V, E)$ be an undirected graph. $V=\{v_1, v_2, \ldots, v_n\}$ is its vertex set, where n is the cardinality of V. $E=\{(v_i, v_j) \subset V\times V\}$ represents its edge set, and $G(S)= (S, E\cap S\times S)$ represents the *subgraph induced* by S, where S is a subset of V. A clique of G is a subset of vertices $C\subseteq V$ such that all pairs of vertices in C are connected by an edge in G, i.e., for all v_i , $v_j\in C$, $(v_i , v_j) \in E$. The maximum clique problem is then to find a clique of maximum cardinality. Note that there may have more than one optimal solution in a MCP instance. A vertex is called *optimal vertex* only if it is contained in any one optimal clique.

Several concepts used in our BBEDA algorithm are introduced as follows.

An *independent set* is a subset $IS\subset V$ such that no any two vertices in IS are joined by an edge in G.

Following the definition of *degree* in paper [9], the *static degree* refers to *degree* in G, and the *dynamic degree* refers to the degree in the induced subgraph $G(S)$.

With respect to a clique C, an *Addible Set* is a subset $Add(C) \subset V \setminus C$ such that all the vertices in $Add(C)$ connects to all the elements in C. A *Level Set* is a subset $Lev(C) \subset V \setminus C$ such that all the vertices in $Lev(C)$ connects to all but one vertex in C.

Greedy adding strategy: it can be described as successively expands C by adding a vertex from $Add(C)$ which has the highest degree in $Add(C)$ until $Add(C)$ is empty. Note that due to $Add(C)$ is changed after every adding step, the vertex selection is based on a *dynamic degree* in $Add(C)$.

B. Feature Analysis and Vertices Partition for MCP

Obviously, except for the single vertex set, an independent set can never appear in a clique, and vice versa. In order to exploiting this property, we partition the vertices set into several independent sets. Denote *LeftV* as the *remained vertex set* that have not been partitioned. The partition of the vertex set could be obtained by using *Greedy adding* strategy in *LeftV* until *LeftV* is empty.

After performing greedy partition, we obtain several independent sets $I = \{I_1, I_2, \cdots, I_M\}$, where M is the number of the independents. Denote $I_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,ni}\}$, where ni is the size of the independent set I_i , then, for any i, j =1, 2, $\cdots$, M and $i\neq j$

$$\begin{cases} I_1 \cup I_2 \cup \ldots \cup I_M = V \\ I_i \cap I_j = \Phi \end{cases} \quad (1)$$

C. Bi-level Estimation of Distribution Model

As discussed above, since each group is an independent set, in a group there is at most one vertex that can appear in an optimal solution. Thus two problems have been put forward. First, which group has higher probability to contain the optimal vertices? Second, which vertex belongs to optimal vertices with high probability within a group? With respect to these two problems, a bi-level blocked probability (BBP) model is constructed to estimate the distribution of the optimal vertices. The BBP model consists of two sub-models, ordinal estimation model and blocked estimation model. The first ordinal estimation model estimates the probability of a group having optimal vertices. The second blocked probability model estimates the relatively probability of a vertex contained in an optimal solution within a group. Next, the two models are discussed, respectively.

D. Ordinal Estimation Model

When the number of groups is larger than the maximum clique size, it means some groups have no optimal vertex. So there exist competition relations among groups. When generating a new point in the solution space, the visiting order of the groups will have a great effect on the final result.

In order to estimate the probability of a group having optimal vertex, *weight* is assigned to every group in G. Denote $W(i)$ as the weight of group I_i . The visiting order of the groups is determined by the value of W. Then W influences the search direction in such a way that the algorithm will first search within the local area consisting of the vertices in the high weight group.

The result of an ordinal estimation is a new visiting

order of the group, thus the *weight* of the permutated groups is in a descending order. Denote $J = [j_1, j_2, \cdots, j_m]$ as a permutation of the group, then its corresponding *weight* can be expressed as $W(J) = [W(j_1), W(j_2), \cdots, W(j_m)]$. The ordinal estimation is to reorder J such that $W(j_1) \geqq W(j_2) \geqq \cdots \geqq W(j_m)$.

The value of W will be updated during the search procedure. The detail updating method is given in section VI.

E. Blocked Probability Model

The purpose of the blocked estimation is to measure the relative probability distribution of optimal vertices within a group.

The problem here is which vertex belongs to an optimal solution with high probability. After a considerable amount of searches, we can obtain a large number of cliques which are uniformly distributed in the search space. If one vertex v occurs in a clique with the size S_c, it is possible that there exists a clique with the size S_c+1 containing the vertex v in G. Moreover, if all the cliques containing v have the size less than a small value s_c, v is unlikely to be in a clique with a size larger than s_c and in an optimal solution.

Following this initialization, $B(v)$ is introduced to record the best clique size containing v. The value of $B(v)$ will be updated during the search procedure. The detail updating method is given in section VI.

Denote $P(v_{ij})$ as the probability of selecting vertex v_{ij}. BBEDA uses the pseudo-Boltzman distribution for selecting promising vertex,

$$P(v_{ij}) = \frac{1}{z_T} Exb\{\frac{f(v_{ij})}{T}\} \tag{2}$$

where T is set to one, z_T is the usual partition function and defined as follows:

$$z_T = \sum_{j=1}^{ni} Exb\{\frac{f(v_{ij})}{T}\}$$

Note that here the '*Exb*' is a parameter which represents the prior knowledge of the distributions of the optimal vertices, and controls the direction of the search. The variable $f(v_{ij})$ is evaluated by the following equation:

$$f(v_{ij}) = \begin{cases} B(V_{ij}) - mg_i & B(V_{ij}) \neq 0 \,\&\, V_{ij} \in Add(c) \\ tcs - mg_i & B(V_{ij}) = 0 \,\&\, V_{ij} \in Add(c) \\ 0 & V_{ij} \notin Add(c) \end{cases} \tag{3}$$

where mg_i is used for normalization and the value of mg_i is defined as the minimum value of $B(v)$ in group I_i except for 0:

$$mg_i = min\ \{\ B(V_{ij})\ \mid\ j = 1,2,\ldots,n_i \text{ and } B(V_{ij}) \neq 0\ \}$$

It can be easily seen that the vertices with B=0 has the privilege when selecting a vertex into clique C. Note that only the vertex has never been visited has B=0. This provides the diversification of the algorithm.

The purpose of *prohibitions* (*vertex penalty*) in DLS_MC is to provide diversification, while our BBEDA can not only provide diversification, but also estimate the promising area in the search space.

F. Sampling from BBP Model

BBP model gives the estimation of the distribution of optimal vertices. By sampling from BBP model, a clique in promising area can be generated. A *vertex–adding* construction is used in sampling, which selects a vertex from group to group in the order J according to a probability distribution in (2).

The cliques generated in this way are all feasible solutions to MCP. Moreover, due to the unused vertex is assigned to a high $f(v_{ij})$ value, new and promising areas of the search space can be explored. The procedure of sampling from BBP model is as follows:

Input: I, W, B;
  Clique=Φ;
  J = reorder group I such that $W(j_1) \geqq W(j_2) \geqq \cdots \geqq W(j_m)$;
  For i=1: M
  Sample a vertex V_{ij} from group $I(j_i)$ according to the probability distribution given by equation (2).
  Clique= *Clique* $\cup v$;
  End For
Output: *Clique*

III. Bi-level Blocked Estimation of Distribution Algorithm for MCP

In this section, the local search methods in BBEDA are described at first. Then the updating method of BBP model during the local search is present. Based on them, the outline of BBEDA algorithm for MCP is given.

A. Local Search

Local search is a well-known technique that can efficiently search for locally optimal solutions. In our novel algorithm, local search is integrated into the BBP model to capture optimal solution. Based on the blocked structure obtained by vertices partition, a new *neighborhood construction* technique is designed to guide the local search towards a promising direction.

Let C be the current clique (or point), its *neighbors* $N(C)$

$\subseteq V$ consists of all the vertices connected to most of vertices in C. Obviously the vertex with a larger number of edges connected to C is more promising to increase the size of C by interchanging with some vertices in C.

Denote $Deg(V_{ij})$ as the dynamic degree in $G(C \cup V_{ij})$. Let $maxDeg(I_i)$ be the maximum value of $Deg(V_{ij})$ for all V_{ij} in group I_i . *General neighborhoods*, denoted by $Ng(C)$, is defined as follows:

$$Ng(C) = \{ V_{ij} \mid V_{ij} \in Add(C) \ \& \ Deg(V_{ij}) = maxDeg(I_i) \}$$

Within group I_i only one vertex could appear in a clique. Using *greedy adding strategy* in $Ng(C)$, a clique $N_S(C)$ can be obtained, which is called the *strong neighborhood* of C.

Thus the local search is to find a maximum clique in the subgraph $G(N_S(C) \cup C)$, which consists of two strongly connected cliques. Though it is solvable by *Hungarian Algorithm* in polynomial time, its performance in local search is worse than the *greedy adding strategy* in solution quality and processing time. Using this strategy, a new clique C_{new} is generated.

In order to avoid unproductive local search, two variables are introduced as termination condition variable. One is *NoUp*, which counts the consecutive iterations with no increase in the clique size. The other is *Cycle*, which is a logical variable indicating the local search cycles.

B. BBP Model Update

BBP model is built to collect information about distributions and give the estimation of promising area in the solution space. In the BBP, the probability model is updated by updating $B(V)$ and $W(J)$.

As discussed above, $B(v)$ is introduced to record the best clique size containing v. Therefore, $B(v)$ is updated every time when a larger size of clique is found. Note that in the greedy-adding procedure, the location of a clique in the search space is determined step by step. The earlier is a vertex added into the clique, the more influence it will have on the final location. Therefore, $B(v)$ is updated only for those vertices added to the clique C in the first half addition steps.

Different from $B(v)$, *weight* $W(J)$ is only updated once just after sampling a new point C. After sampling a new point C from BBP model, if a group I_i has no vertex in C but has vertices in $Lev(C)$, it is likely to put a vertex into C just when change the sequence J. Therefore, the weights of such groups are increment by one, i.e., for the group $I_i \cap C = \Phi$ and $I_i \cap Lev(C) \neq \Phi$,

$$W(i) = W(i)+1.$$

C. Bi-level Blocked Estimation of Distribution Algorithm

The new BBEDA algorithm combines EDAs and local search technique. BBEDA is history-sensitive. The global information is collected during local search and used in generating new points. The new point is sampled from the BBP model and locates the start point of next local search. The whole procedure alternates between the phase of generating a new point and of the local search from this new point. The outline of BBEDA is given below:

```
Input: target size, maxSteps, maxNoUp, Exb;
Vertex partition;
Initialization: steps = 0, optimal size = 0, Cycle = 0;
                W(j) = 0, B(v) = 0;
While optimal size < target size & steps < maxSteps
    C=Sample from BBP model;
    While NoUp < maxNoUp & Cycle < 1
        N_S(C) = Neighborhood Construction for C;
        C_new = greedy adding in G(N_S(C) ∪ C);
        Update W(j) , B(v), NoUp and Cycle;
        steps = steps + size of C + size of N_S(C);
    End While
End While
Output: optimal size, steps
```

IV. Simulation Results

In order to evaluate the performance and behavior of BBEDA, we performed extensive computational experiments on all MAX-CLIQUE instances from the Second DIMACS Implementation Challenge, which have also been used extensively for benchmarking purposes in the recent literature on MAX-CLIQUE algorithms. The 80 DIMACS MAX-CLIQUE instances were generated from problems in coding theory, fault diagnosis problems, Keller's conjecture on tilings using hypercubes and the Steiner triple problem, and so on.

All the experiments were implemented using lcc compiler in Matlab 6.5 *lcc* complier on a computer with a P4 2.8GHz CPU and 256 MB RAM.

A. BBEDA performance

Due to the random elements of BBEDA, we performed 100 independent runs for each instance. In all cases, we set *maxNoUp* = 3, *maxSteps* = 100,000,000, and target clique sizes (*tcs*) to the best clique sizes we could find in literature.

Table 1 (due to large size, Table 1 is placed at the end of this paper) shows the experimental results including:

- Solution Quality: it is introduced by Silvia Richter to measure the solution quality of vertex cover [8]. It is denoted as a triple *a*-*b*-*c*, where *a* is the number of runs (out of 100) in which the algorithm found a clique with the size *tcs*; *b* is the number of runs in which a clique of size *tcs* was not found, but a clique of size *tcs*-1 was found; and *c* is the number of runs where only a clique of size *tcs*-2 was found.
- Steps: the number of vertices added to the clique, averaged over all successful runs. With respect to those instances where the target clique size was not reached, the

TABLE 1
COMPARISON OF BBEDA AND DLS_MC ON DIMACS BENCHMARK GRAPHS

Instance	*MaxNoUp*	*Exb*	BR	BBEDA			DLS_MC		
				Quality	Steps	CPU(s)	Quality	Steps	CPU(s)
san200_0.7_1	*5*	*2*	*30*	*100-0-0*	979	0.16	*100-0-0*	1727	0.01
san200_0.7_2	*5*	*2*	*18*	*100-0-0*	953	0.14	*100-0-0*	33661	0.07
san200_0.9_1	*5*	*2*	*70*	*100-0-0*	329	0.08	*100-0-0*	415	0.01
san200_0.9_2	*5*	*2*	*60*	*100-0-0*	222	0.05	*100-0-0*	347	0.01
san200_0.9_3	*5*	*2*	*44*	*100-0-0*	140	0.03	*100-0-0*	1564	0.01
san400_0.5_1	*5*	*2*	*13*	*100-0-0*	963	0.14	*100-0-0*	26235	0.16
san400_0.7_1	*5*	*2*	*40*	*100-0-0*	5054	1.27	*100-0-0*	29635	0.11
san400_0.7_2	*5*	*2*	*30*	*100-0-0*	4246	0.88	*100-0-0*	57358	0.21
san400_0.7_3	*5*	*2*	*22*	*100-0-0*	17138	2.28	*100-0-0*	113905	0.42
san400_0.9_1	*5*	*2*	*100*	*100-0-0*	376	0.13	*100-0-0*	1820	0.01
san1000	*5*	*2*	*15*	*100-0-0*	106053	31.36	*100-0-0*	521086	8.36
sanr200_0.7	*5*	*2*	*18*	*100-0-0*	66	0.02	*100-0-0*	1342	0.01
sanr200_0.9	*5*	*2*	*42*	*100-0-0*	939	0.19	*100-0-0*	15739	0.01
sanr400_0.5	*5*	*2*	*13*	*100-0-0*	4801	0.63	*100-0-0*	9918	0.04
sanr400_0.7	*5*	*2*	*21*	*100-0-0*	268	0.05	*100-0-0*	8475	0.02
p_hat300-1	*5*	*2*	*8*	*100-0-0*	56	0.02	*100-0-0*	133	0.01
p_hat300-2	*5*	*2*	*25*	*100-0-0*	46	0.02	*100-0-0*	87	0.01
p_hat300-3	*5*	*2*	*36*	*100-0-0*	272	0.06	*100-0-0*	476	0.01
p_hat500-1	*5*	*2*	*9*	*100-0-0*	68	0.02	*100-0-0*	114	0.01
p_hat500-2	*5*	*2*	*36*	*100-0-0*	98	0.02	*100-0-0*	200	0.01
p_hat500-3	*5*	*2*	*50*	*100-0-0*	183	0.06	*100-0-0*	1075	0.01
p_hat700-1	*5*	*2*	*11*	*100-0-0*	109	0.03	*100-0-0*	1767	0.02
p_hat700-2	*5*	*2*	*44*	*100-0-0*	78	0.02	*100-0-0*	251	0.01
p_hat700-3	*5*	*2*	*62*	*100-0-0*	177	0.06	*100-0-0*	525	0.01
p_hat1000-1	*5*	*2*	*10*	*100-0-0*	18	0.01	*100-0-0*	230	0.01
p_hat1000-2	*5*	*2*	*46*	*100-0-0*	87	0.03	*100-0-0*	415	0.01
p_hat1500-1	*5*	*0.7*	*12*	*100-0-0*	119289	39.36	*100-0-0*	126872	2.70
p_hat1500-2	*5*	*0.7*	*65*	*100-0-0*	1185	0.56	*100-0-0*	730	0.01
p_hat1500-3	*5*	*0.7*	*94*	*100-0-0*	8345	6.39	*100-0-0*	1828	0.01
keller4	*5*	*2*	*11*	*100-0-0*	21	0.02	*100-0-0*	31	0.01
keller5	*5*	*2*	*27*	*100-0-0*	2720	0.49	*100-0-0*	4067	0.02
MANN_a9	*5*	*2*	*16*	*100-0-0*	41	0.02	*100-0-0*	21	0.01
MANN_a27	*5*	*2*	*126*	*100-0-0*	2614	1.70	*100-0-0*	41976	0.05

reported statistics are over runs where a clique with size *tcs*-1 was found, or over runs where a clique with size *tcs*-2 was found even if a clique with the size *tcs*-1 was not found. For purpose of distinction, the statistics in the latter two cases are shown in parentheses;

• CPU(s): the expense of CPU times (in seconds), averaged over all successful runs. With respect to those instances where the target clique size was not reached, the statistics are defined as above;

B. Comparison With DLS_MC

DLS_MC is the state of the art algorithm for MCP at present. It explicitly aims at exploring much more solution space between explore and exploit strategy. It performs local search by a simple one-vertex swapping operator and frequently restarts from a new initial point after a very short duration of local search. The diversification in DLS_MC is provided by a strict prohibition mechanism, which is realized by *vertex penalties* that is proportional to the vertices visiting time.

Table 1 also gives the simulation results of DLS_MC. The DLS_MC was also run 100 times with the same limit on steps as BBEDA. As can be seen from Table 1, the running time of BBEDA is longer than that of DLS_MC, while the number of steps of BBEDA is less than that of DLS_MC. Since the two algorithms were implemented by different computer languages, the running time can only be roughly compared. However the *step* in two algorithms can accurately represent vertex adding or deleting, which allows direct comparison between these two algorithms. BBEDA needs more computational time in doing distribution estimation and neighborhood construction, but this extra time can be compensated by decreasing the number of unproductive searching steps.

C. Comparison with EA/G

EA/G is based on the assumption that good solutions have similar structure. It cannot achieve good performance when this assumption is not valid. For example, the

TABLE 2
COMPARISON OF BBEDA AND EA/G ON SOME DIMACS BENCHMARK GRAPHS

Instance	*MaxNo Up*	BBEDA			EA/G		
		Avg.	Best	CPU(s)	Avg.	Best	CPU(s)
brock200_4	*10*	*17*	17	3.70	16.5	17	2.0
brock400_2	*10*	*29*	29	636.21	24.7	25	3.7
brock400_4	*10*	*33*	33	38.15	25.1	33	3.9
brock800_2	*10*	*22.6*	*24*	1622.8	20.1	21	8.9
brock800_4	*10*	*26*	*26*	639.94	19.9	21	8.9
C500.9	*10*	*57*	*57*	67.1	55.2	56	5.7
C1000.9	*10*	*68*	*68*	279.5	64.4	67	21.2
C2000.5	*50*	*16*	*16*	69.65	14.9	16	28.6
C2000.9	*50*	*76.7*	*78*	1729.6	70.9	72	45.2

TABLE 3
THE CPU(S) ON BROCK200_2 AT DIFFERENT VALUE OF EXB

Exb	0.3	0.5	0.8	1.0	1.5	2.0	2.5
CPU(s)	*5.92*	4.27	2.48	6.29	8.68	10.85	9.16
Steps	*58611*	41520	23933	59676	85324	106802	90998

proximate optimality is not significant in DIMACS, such as *C families* and *brock families*. Table 2 gives the simulation results for two algorithms on these instances. Other than different implementation languages, note that the *best* value of EA/G is the best result in 10 runs, while the *CPU*(*s*) value is the run time for one run. So the time superiority of EA/G than BBEDA does not mean that the BEDDA is worse than the EA/G in computational performance. As can be seen from Table 2, compared to EA/G, BBEDA can always find a larger clique. It shows that BBEDA has a higher flexibility on the instances that the proximate optimality feature is not significant.

D. Effects of Parameter Exb

BBEDA contains a single parameter "*Exb*", which represents the prior knowledge on the distribution of the optimal solutions.

To assess the effects of *Exb* on the performance of BBEDA, BBEDA is tested on the DIMACS benchmark problem *brock*200_2. Table 3 lists the corresponding data of the average size of the largest cliques found in 100 independent runs for different parameter settings of *Exb*.

As can be seen from Table 3, although at different value of *Exb* the algorithm can finally find the optimal solution, the appropriate settings of *Exb* can significantly accelerate the searching speed. It is proved again that proper domain knowledge is helpful in stochastic search algorithm.

V. CONCLUSIONS

In this study, a Bi-level Blocked Estimation of Distribution Algorithm with local search is proposed for Maximum Clique Problem. By vertices partition, a Bi-level Blocked Probability Model was built, which can collect the information from local search and estimate the distribution of optimal solutions. The BBEDA achieves the relationships utilizing in a bivariate probability model at the computational cost of univariate probability model. The simulations on extensive DIMACS benchmark instances confirmed that the BBEDA is an inexpensive, high-quality heuristic solution technique for Maximum Clique Problems.

REFERENCES

[1] J.Hastad, "Clique is hard to approximate within $n^{1-\varepsilon}$, " *Acta Mathematica*, vol.182, pp.105-142, 1999

[2] You mei Li and Zongben Xu, "An ant colony optimization heuristic for solving maximum independent set problems", *Proc.2003 5th International Conference on Computational Intelligence and Multimedia Applications,* vol.3, pp.1663-1666, Aug. 2000.

[3] K.Park, B.Carter, "on the effectiveness of genetic search in combinatorial optimization", Technique Report , 1994.

[4] Qingfu Zhang, Jianyong Sun and Edward Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem", IEEE Trans. Evol. Comput., vol.9, pp. 192-200, 2005

[5] Hoos, H. H., and Stutzle,T. , "Stochastic Local Search: Foundations and Applications", Morgan Kaufman Publishers, USA., 2000

[6] Wayne Pullan and Holger H. Hoos, "Dynamic local search for the maximum clique problem", *Journal of Artificial Intelligence Research*, vol.25, pp.159-185, 2006

[7] Roberto Battiti, "Reactive local search for Maximum Clique", *Journal of Scheduling*, vol.7, pp.7-34, 2004

[8] Silvia Richter, Malte Helmert and Charles Gretton, "A stochastic local search to vertex cover", *Proc.2007, Australian Conference on Artificial Intelligence,* pp.786–789, 2007.