Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 5064–5071
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A simple simulated annealing algorithm for the maximum clique problem

Xiutang Geng *, Jin Xu, Jianhua Xiao, Linqiang Pan

Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Received 20 June 2006; received in revised form 3 June 2007; accepted 6 June 2007

Abstract

Simulated annealing technique has mostly been used to solve various optimization and learning problems, and it is well known that the maximum clique problem is one of the most studied NP-hard optimization problems owing to its numerous applications. In this note, a simple simulated annealing algorithm for the maximum clique problem is proposed and tested on all 80 DIMACS maximum clique instances. Although it is simple, the proposed simulated annealing algorithm is efficient on most of the DIMACS maximum clique instances. The simulation results show that the proposed simulated annealing algorithm outperforms a recent efficient simulated annealing algorithm proposed by Xu and Ma, and the solutions obtained by the proposed simulated annealing algorithm have the equal quality with those obtained by a recent trust region heuristic algorithm of Stanislav Busygin.

Keywords: Simulated annealing algorithm; Maximum clique problem; Minimum vertex cover problem; NP-hard optimization problem; Heuristic algorithm

1. Introduction

In this note, only simple undirected graphs are considered. For notions and notations on graph theory and simulated annealing, please refer to [21,19], respectively.

For a given graph, the maximum clique problem is to find the largest number of vertices, any two of which are adjacent. In 1999, Hastad [9] showed that unless NP = ZPP, for any $\varepsilon > 0$, there is no polynomial-time algorithm to approximate Max-Clique within a factor $n^{1-\varepsilon}$, where ZPP is the class of languages which can be recognized by randomized algorithms for which the expected running time is polynomial in the input size. So it seems to be impossible to have an exact polynomial time algorithm for the maximum clique problem, and approximation of large clique is also hard. But, the maximum clique problem is related to many real-life

* Corresponding author. Tel.: +86 27 87543563; fax: +86 27 87543130.
E-mail addresses: gxt1028@163.com (X. Geng), jxu@mail.hust.edu.cn (J. Xu), jhxiao@smail.hust.edu.cn (J. Xiao), lqpan@mail.hust.edu.cn (L. Pan).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2007.06.009

problems, such as classification theory, coding theory and project selection. Hence there is, in practice, a great need of efficient heuristic algorithms.

Some exact algorithms for the maximum clique problem have been proposed. For instance, Balas and Yu [1] discussed how to find a maximum clique in an arbitrary graph; Carraghan and Pardalos [5] gave an exact algorithm for the maximum clique problem; Pardalos and Rodger [15] proposed a branch and bound algorithm for the maximum clique problem; and Östergård [14] introduced an improved branch and bound algorithm for the maximum clique problem.

However, these exact algorithms are only adapted to some graphs with small size. So, various approximate methods have been applied in solving the maximum clique problem. For example, Zhang et al. [23] proposed an algorithm for the maximum clique problem of a given graph based on the Hopfield networks; Bomze et al. [3] presented a new heuristic algorithm for the maximum clique problem based on annealed replication; Galán Marín et al. [7] discussed the modeling competitive Hopfield networks for the maximum clique problem; Hansen et al. [8] proposed a basic variable neighborhood search for the maximum clique problem; Zhang et al. [22] described an evolutionary algorithm with guided mutation for the maximum clique problem; Katayama et al. [10] proposed an effective local search for the maximum clique problem; Busygin [4] developed a new trust region technique for the maximum weight clique problem; Dukanovic and Rendl [6] presented a new method of semi-definite programming relaxations for maximal clique problem; Tomita and Kameda [18] proposed an efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. For the above various approximate algorithms, different algorithms are efficient for different types of DIMACS maximum clique instances. For instance, the algorithm in [22] is more efficient for the type *gen* maximum clique instances than the algorithm in [4], while the algorithm in [4] is more efficient for the type *brock* maximum clique instances than the algorithm in [22]. As a result, a new approximate algorithm for the maximum clique problem is of interest.

Simulated annealing technique was formally introduced for solving combinatorial optimization problems [11] in 1983. Since then, it has been widely used to solve many optimization problems: Bandyopadhyay et al. [2] described a method for finding decision boundaries using simulated annealing algorithm; Sánchez et al. [17] developed a simulated annealing-based method for inducting both parameters and structure of a fuzzy classifier; Poranen [16] presented a simulated annealing algorithm for determining the thickness of a graph; Xu et al. [20] considered an efficient simulated annealing algorithm for the minimum vertex cover problem; Lin and Chen [12] proposed a simulated annealing algorithm based on graph model to solve multi-layer constrained via minimization; Nalini and Raghavendra Rao [13] discussed a systematic study of efficient heuristic in the successful attacks of some block ciphers by simulated annealing algorithm. Simulated annealing is a choice technique for finding good solutions to a wide variety of problems. However, any published paper for the maximum clique problem based on simulated annealing algorithm hasn't been proposed yet. Hence, an efficient simulated annealing algorithm for the maximum clique problem is of interest and necessity.

For a given graph with n vertices, in this note, a simple simulated annealing algorithm for finding a clique m (maximum clique or approximate maximum clique) is proposed, where $n \geqslant m$. With the proposed simulated annealing algorithm, a clique of the given graph can be found. Computational experiments indicate that the proposed simulated annealing algorithm is more efficient than a recent efficient simulated annealing algorithm in [20], and the solutions obtained by the proposed simulated annealing algorithm have the equal quality with those obtained by a recent trust region heuristic algorithm in [4].

The rest of the note is organized as follows. Section 2 presents the objective function of the maximum clique problem. The simple simulated annealing algorithm for the maximum clique problem is described in Section 3. Some experimental results obtained are presented in Section 4. The note is concluded in Section 5.

2. The objective function for the maximum clique problem

To solve the maximum clique problem with different methods, different objective functions have to be set up. In this note, a new objective function for the maximum clique problem is proposed based on simulated annealing algorithm, and the establishment of the objective function is as follows.

Let $G(V,E)$ be a graph with n vertices, and let $A_G(a_{k,l})$ be the adjacency matrix of $G(V,E)$, where k, $l = 0, 1, \ldots, n-1$. Then $G(V,E)$ has a clique of size m if there exists a permutation σ of V such that

$$\sum_{k=0}^{m-2}\sum_{l=k+1}^{m-1}(1-a_{\sigma(k),\sigma(l)})=0. \tag{1}$$

So, the objective of the maximum clique problem is to minimize the left of formula (1) equal to zero by trying random variation of permutation σ. The random variation of permutation σ is achieved by swapping the labels of two different vertices of $G(V,E)$, and therefore transforms the adjacency matrix $A_G(a_{k,l})$. The rule of choosing two different vertices for swapping is described in step 3 of Section 3. In this note, the objective function for the maximum clique problem can be described as follows:

$$F(G,\sigma)=\sum_{k=0}^{m-2}\sum_{l=k+1}^{m-1}(1-a_{\sigma(k),\sigma(l)}). \tag{2}$$

3. The simple simulated annealing algorithm for the maximum clique problem

Simulated annealing technique can find a good solution to an optimization problem by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. The slower the cooling schedule, the more likely the algorithm is to find an optimal or near-optimal solution.

Suppose the initial state (initial solution) of the maximum clique problem is defined by a random initial permutation σ of V in $G(V,E)$. After the labels of two different vertices of $G(V,E)$ swap each other, a new permutation σ' (new solution) of V is generated. In this note, the process of swap is called state transition, and the new permutation σ' is a neighbor to the initial permutation σ. Hence the increment of objective functions is described as follows:

$$\Delta F=F(G,\sigma')-F(G,\sigma). \tag{3}$$

For simulated annealing algorithm, if $\Delta F\leqslant 0$, the new permutation σ' is accepted. Otherwise the probability of accepting the new permutation σ' is controlled by the acceptance function, and the rule of accepting the new permutation σ' is described in step 5 of the proposed simulated annealing algorithm's procedures. Where the acceptance function can be described as follows:

$$p=\mathrm{e}^{-\Delta F/t}, \tag{4}$$

where t denotes current temperature.

Finally, the detailed procedures of the proposed simulated annealing algorithm for the maximum clique problem are given as follows:

Step 1: Initialize parameters.
Set initial temperature T_{I}, end temperature T_{S} and cooling coefficient α of temperature. Input the adjacency matrix A_G $(a_{k,l})$ of a DIMACS maximum clique instance and a positive integer m which denotes the size of a clique of $G(V,E)$, and compute objective function $F(G,\sigma)$. Suppose initial permutation σ $(i)=i$, $i=0,1,\ldots,n-1$.

Step 2: Adjust the initial permutation σ.
Intuitively, a vertex with higher degree (the number of incident edges) is more probable to be included in G's a maximum clique. So the permutation σ is adjusted with respect to the degrees of vertices so that $d(\sigma(1))<d(\sigma(2))<\cdots<d(\sigma(n))$, where $d(i)$ denotes the degree of the vertex i.

Step 3: Select two different vertices v_u and v_w of $G(V,E)$.
In the note, for the random variations of permutation σ, the selection of two different vertices v_u and v_w is crucial, where $u=0,1,2,\ldots,m-1$, and $w=m,m+1,\ldots,n-1$. In the proposed simulated annealing algorithm, two restricting conditions for selecting v_u and v_w are set to help search a maximum clique or an approximate maximum clique more efficiently. Let

$$F'(G,v_z)=\sum_{i=0,i\neq z}^{m-1}a_{i,z}. \tag{5}$$

Then the two restriction conditions are described as follows:
Restricting condition 1:
If $F'(G,v_u) \leqslant F'(G,v_w)$, the two vertices v_u and v_w are accepted, and go to step 4.
Restricting condition 2:
If $F'(G,v_u) > F'(G,v_w)$, go back to step 3 and select a different vertex for v_w. But, if $F'(G,v_u) > F'(G,v_w)$ appears for more than $8n$ times, the two vertices v_u and v_w are accepted, and go to step 4.

Step 4: Perform state transition.
After the selection of the two different vertices v_u and v_w of $G(V,E)$, swap their labels and perform the state transition or the varying of permutation σ, and a new permutation σ' is attained. Finally, compute objective function $F(G,\sigma')$. If $F(G,\sigma') = 0$, end the proposed simulated annealing algorithm; otherwise go to step 5.

Step 5: Accept or reject the new permutation σ'.
Compute the increment of the objective functions $\Delta F = F(G,\sigma') - F(G,\sigma)$. If $\Delta F \leqslant 0$, $\sigma = \sigma'$, i.e. σ' is accepted. Otherwise compute the probability of accepting the new permutation σ' according to formula (4), and generate a random number β, where $\beta \in [0,1]$. If $p > \beta$, $\sigma = \sigma'$; otherwise the new permutation σ' is rejected.

Step 6: Continue or end the proposed simulated annealing algorithm.
Compute current temperature $t = \alpha t$. If $t < T_S$, end the proposed simulated annealing algorithm; otherwise go back to step 3 and select two different vertices v_u and v_w.

4. Simulation results

In order to assess the effectiveness of the proposed simple simulated annealing algorithm, in this note, simulation was carried out on all 80 DIMACS maximum clique instances downloaded from the URL http://cs.hbg.psu.edu/txn131/INSTANCES/clique.html. The parameters used in the proposed simulated annealing

Table 1
Parameters used in the proposed simulated annealing algorithm

Parameters	Initial value	Meaning
T_I	100.0	Initial temperature
T_S	0.001	End temperature
α_1	0.9996	The constant to decrease temperature for Table 2
α_2	0.9995	The constant to decrease temperature for Table 3
L	$8n$	The maximum times for rejecting the selected vertex v_u and v_w

Table 2
The simulation results of the proposed simulated annealing algorithm (short for SAA) with $\alpha_1 = 0.9996$ were compared with those obtained by a recent efficient simulated annealing algorithm (short for ESA) for the minimum vertex cover problem in [20] on 9 DIMACS maximum clique instances

Instance	Order	Density	Clique	cover	ESA [20]		Proposed SAA	
					Time (s)	Rate (%)	Time (s)	Rate (%)
MANN_a9	45	0.927	16	29	–	100	<1	100
c-fat200-2	200	0.163	24	176	–	100	<1	100
c-fat500-1	500	0.036	14	486	–	98	<1	100
hamming6-2	64	0.905	32	32	–	100	<1	100
johnson8-2-4	28	0.556	4	24	–	100	<1	100
johnson32-2-4	496	0.879	16	480	–	99	<1	100
p_hat300-3	300	0.744	36	264	–	98	2	100
p_hat500-1	500	0.253	9	491	–	99	<1	100
sanr200_0.7	200	0.697	18	182	–	98	<1	100

Where notation "–" denotes that the computation time of the algorithm in [20] is absent.

Table 3
The simulation results of the proposed simulated annealing algorithm with $\alpha_2 = 0.9995$ were compared with those obtained by a trust region technique (short for TR) for the maximum weight clique problem in [4] on all 80 DIMACS maximum clique instances

Instance	Order	Density	Clique	TR [4]		Proposed SAA			
				|C|	Time (s)	|C|	Time (s)	Rate (%)	Run times
brock200_1	200	0.745	21	21	1	·21	5	70	20
brock200_2	200	0.496	12	12	<1	·12	8	10	20
brock200_3	200	0.605	15	15	1	14	2	95	20
brock200_4	200	0.658	17	17	<1	16	<1	100	20
brock400_1	400	0.748	27	27	4	25	22	20	20
brock400_2	400	0.749	29	29	4	25	29	30	20
brock400_3	400	0.748	31	31	5	25	26	35	20
brock400_4	400	0.749	33	33	4	25	26	35	20
brock800_1	800	0.649	23	23	36	21	131	10	20
brock800_2	800	0.651	24	24	35	21	124	10	20
brock800_3	800	0.649	25	25	37	21	122	10	20
brock800_4	800	0.650	26	26	35	21	125	10	20
C125.9	125	0.898	34	34	<1	·34	<1	100	20
C250.9	250	0.899	44	44	1	·44	4	95	20
C500.9	500	0.900	⩾57	55	8	·57	59	5	20
C1000.9	1000	0.901	⩾68	64	71	·68	222	0.5	200
C2000.5	2000	0.500	⩾16	16	1547	·16	877	10	20
C2000.9	2000	0.900	⩾77	72	1519	74	776	20	20
C4000.5	4000	0.500	⩾18	17	15,558	17	903	20	20
c-fat200-1	200	0.077	12	12	<1	·12	<1	100	20
c-fat200-2	200	0.163	24	24	<1	·24	<1	100	20
c-fat200-5	200	0.426	58	58	<1	·58	<1	100	20
c-fat500-1	500	0.036	14	14	4	·14	4	95	20
c-fat500-2	500	0.073	26	26	3	·26	<1	100	20
c-fat500-5	500	0.186	64	64	2	·64	<1	100	20
c-fat500-10	500	0.374	126	126	3	·126	<1	100	20
DSJC500.5	500	0.500	⩾13	13	5	·13	17	95	20
DSJC1000.5	1000	0.500	⩾15	14	36	·15	363	15	20
gen200_p0.9_44	200	0.900	44	42	<1	·44	21	55	20
gen200_p0.9_55	200	0.900	55	55	1	·55	1	100	20
gen400_p0.9_55	400	0.900	55	51	2	·55	31	10	20
gen400_p0.9_65	400	0.900	65	65	2	·65	28	90	20
gen400_p0.9_75	400	0.900	75	75	2	·75	75	35	20
hamming6-2	64	0.905	32	32	<1	·32	<1	100	20
hamming6-4	64	0.349	4	4	<1	·4	<1	100	20
hamming8-2	256	0.969	128	128	<1	·128	3	95	20
hamming8-4	256	0.639	16	16	1	·16	<1	100	20
hamming10-2	1024	0.990	512	512	38	·512	427	55	20
hamming10-4	1024	0.829	⩾40	36	45	·40	144	80	20
johnson8-2-4	28	0.556	4	4	<1	·4	<1	100	20
johnson8-4-4	70	0.768	14	14	<1	·14	<1	100	20
johnson16-2-4	120	0.765	8	8	<1	·8	<1	100	20
johnson32-2-4	496	0.879	16	16	5	·16	<1	100	20
keller4	171	0.649	11	11	1	·11	<1	95	20
keller5	776	0.751	27	26	16	·27	143	4	100
keller6	3361	0.818	⩾59	53	1291	51	644	10	20
MANN_a9	45	0.927	16	16	<1	·16	<1	100	20
MANN_a27	378	0.990	126	125	1	·126	49	1	100
MANN_a45	1035	0.996	345	342	17	334	393	20	20
MANN_a81	3321	0.999	⩾1100	1096	477	1080	1879	100	20
p_hat300-1	300	0.244	8	8	1	·8	<1	100	20
p_hat300-2	300	0.489	25	25	1	·25	<1	100	20
p_hat300-3	300	0.744	36	35	1	·36	2	100	20
p_hat500-1	500	0.253	9	9	3	·9	<1	100	20
p_hat500-2	500	0.505	36	36	4	·36	1	100	20

(*continued on next page*)

Table 3 (*continued*)

Instance	Order	Density	Clique	TR [4]		Proposed SAA			
				|C|	Time (s)	|C|	Time (s)	Rate (%)	Run times
p_hat500-3	500	0.752	⩾50	48	4	·50	32	75	20
p_hat700-1	700	0.249	11	11	10	·11	18	95	20
p_hat700-2	700	0.498	44	44	12	·44	3	100	20
p_hat700-3	700	0.748	⩾62	62	11	·62	12	100	20
p_hat1000-1	1000	0.245	⩾10	10	28	·10	6	100	20
p_hat1000-2	1000	0.490	⩾46	45	34	·46	16	100	20
p_hat1000-3	1000	0.744	⩾68	65	32	·68	100	80	20
p_hat1500-1	1500	0.253	12	12	95	·12	490	1	100
p_hat1500-2	1500	0.506	⩾65	64	111	·65	40	100	20
p_hat1500-3	1500	0.754	⩾94	91	108	·94	215	70	20
san200_0.7_1	200	0.700	30	30	1	17	9	10	20
san200_0.7_2	200	0.700	18	18	<1	15	9	10	20
san200_0.9_1	200	0.900	70	70	<1	61	12	5	20
san200_0.9_2	200	0.900	60	60	1	·60	12	5	20
san200_0.9_3	200	0.900	44	40	<1	·44	6	55	20
san400_0.5_1	400	0.500	13	13	2	7	<1	100	20
san400_0.7_1	400	0.700	40	40	3	21	36	5	20
san400_0.7_2	400	0.700	30	30	2	16	25	70	20
san400_0.7_3	400	0.700	22	18	2	17	30	55	20
san400_0.9_1	400	0.900	100	100	2	57	38	10	20
san1000	1000	0.502	15	15	25	8	<1	100	20
sanr200_0.7	200	0.697	18	18	1	·18	<1	100	20
sanr200_0.9	200	0.898	42	41	<1	·42	5	75	20
sanr400_0.5	400	0.501	13	13	2	·13	17	70	20
sanr400_0.7	400	0.700	⩾21	20	2	·21	13	80	20

Where notation "·" denotes that the found clique is exact or best known solution.

algorithm are shown in Table 1. The proposed simulated annealing algorithm was carried out on an AOPEN-PC (PentiumIII, 866 MHz) with C++. In Table 2, the results with $\alpha_1 = 0.9996$ were compared with those obtained by a recent efficient simulated annealing algorithm in [20], and in Table 3, the results with $\alpha_2 = 0.9995$ were compared with those obtained by a trust region algorithm in [4].

For a given graph $G(V,E)$ with n vertices, let $\overline{G(V,E)}$ be the complement of graph $G(V,E)$, let $\omega(G)$ be the size of the maximum clique size of graph $G(V,E)$, and let $c(G)$ be the minimum vertex cover size of graph $\overline{G(V,E)}$, then $\omega(G) = n - c(G)$. So, the maximum clique problem is computationally equivalent to the minimum vertex cover problem.

Therefore, in Table 2, the results of the proposed simulated annealing algorithm were compared with that obtained by a recent efficient simulated annealing algorithm for the minimum vertex cover problem in [20] on 9 DIMACS maximum clique instances, which were tested as example. The proposed simulated annealing algorithm was run 100 times from different initial solution states for each DIMACS maximum clique instance. In Table 2, it can be seen that the simulation results show that the proposed simulated annealing algorithm can find the optimal solutions of the 9 DIMACS maximum clique instances with a rate of 100%, and the average computation time is less than 2 s. However, the algorithm in [20] can only find the optimal solutions with a rate of near 100%.

In Table 3, all 80 DIMACS maximum clique instances are tested with the proposed simulated annealing algorithm. The proposed simulated annealing algorithm was run at least 20 times from different initial solutions for each DIMACS maximum clique instance, and the results of the proposed simulated annealing algorithm were compared with those obtained by a recent trust region technique for the maximum weight clique problem in [4]. As shown in Table 3, the trust region algorithm is faster than the proposed simulated annealing algorithm for some small instances, but for the higher order instance C4000.5 and the equal clique size 17, the proposed simulated annealing algorithm is faster. Exact or best known solutions were found by the trust region algorithm in 57 instances, and the proposed simulated annealing algorithm only found the exact or best known solutions in 56 instances. While the trust region algorithm managed to find more exact solutions (in 57

instances) than the proposed simulated annealing algorithm (56 instances), the results are still encouraging. The proposed simulated annealing algorithm performed very well for the 21 DIMACS maximum clique instances of type *gen* and *p-hat*. For example, the proposed simulated annealing algorithm can find the exact or best known solutions for the 21 instances, but, the trust region algorithm in [4] can only find the exact or best known solutions for 12 instances.

5. Conclusion

In this note, a simple simulated annealing algorithm for the maximum clique problem is proposed. In the proposed simulated annealing algorithm, two restriction conditions for the selection of two different vertices v_u and v_w are given in step 3 of Section 3, the two restriction conditions have improved the efficiency of the proposed simulated annealing algorithm. For all 80 DIMACS maximum clique instances, the proposed simulated annealing algorithm can find an exact or best solution in 56 instances with a rate of 70%. At the same time, the proposed simulated annealing algorithm is efficient for the problems of c-fat, DSJC, gen, hamming, Johnson and p-hat according to Table 3. The simulation results show that the proposed simulated annealing algorithm outperforms a recent efficient simulated annealing algorithm in [20], and the solutions have the equal quality with those obtained by a recent trust region heuristic algorithm of Stanislav Busygin in [4]. Furthermore, the proposed simulated annealing algorithm is also efficient in finding the maximum wheel, the maximum circle, the maximum path and any sub-graph for a random graph.

We focus mainly on simulated annealing algorithm for the maximum clique problem and do not mention on the other heuristic algorithms in this note, such as genetic algorithm and tabu search. Therefore, it may be a subject of future work. One hybrid heuristic algorithm for the maximum clique problem may be considered to find better solutions. Another study in the future may develop a more efficient method for the selection of the above two different vertices v_u and v_w to improve the speed of constringency.

Acknowledgements

The authors thank the anonymous reviewer and Professor Witold Pedrycz for their constructive comments and suggestions. Mr. Ho Simon Wang at Academic Writing Center, HUST, has provided tutorial support to improve the manuscript. The research was supported in part by the National Natural Science Foundation of China (Grant Nos. 60373089, 60674106, and 60533010), the Program for New Century Excellent Talents in University (NCET-05-0612), the Ph.D. Programs Foundation of Ministry of Education of China (20060487014), and the Chenguang Program of Wuhan (200750731262).

References

[1] E. Balas, C.S. Yu, Finding a maximum clique in an arbitrary graph, SIAM Journal on Computing 15 (4) (1986) 1054–1068.
[2] S. Bandyopadhyay, S.K. Pal, C.A. Murthy, Simulated annealing based pattern classification, Information Sciences 109 (1998) 165–184.
[3] I.M. Bomze, M. Budinich, M. Pelillo, C. Rossi, Annealed replication: a new heuristic for the maximum clique problem, Discrete Applied Mathematics 121 (2002) 27–49.
[4] S. Busygin, A new trust region technique for the maximum weight clique problem, Discrete Applied Mathematics 154 (2006) 2080–2096.
[5] R. Carraghan, P.M. Pardalos, An exact algorithm for the maximum clique problem, Operations Research Letters 9 (1990) 375–382.
[6] I. Dukanovic, F. Rendl, Semi-definite programming relaxations for graph coloring and maximal clique problems, Mathematical Programming, Series B 109 (2007) 345–365.
[7] G. Galán-Marín, E. Mérida-Casermeiro, J. Muñoz-Pérez, Modelling competitive Hopfield networks for the maximum clique problem, Computers and Operations Research 30 (2003) 603–624.
[8] P. Hansen, N. Mladenovic, D. Uroševic, Variable neighborhood search for the maximum clique, Discrete Applied Mathematics 145 (2004) 117–125.
[9] J. Hastad, Clique is hard to approximate within $n^{1-\varepsilon}$, Acta Mathematica 182 (1999) 105–142.
[10] K. Katayama, A. Hamamoto, H. Narihisa, An effective local search for the maximum clique problem, Information Processing Letters 95 (2005) 503–511.
[11] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[12] R.B. Lin, S.Y. Chen, Conjugate conflict continuation graphs for multi-layer constrained via minimization, Information Sciences 177 (2007) 2436–2447.
[13] N. Nalini, G. Raghavendra Rao, Attacks of simple block ciphers via efficient heuristics, Information Sciences 177 (2007) 2553–2569.
[14] P.R.J. Östergård, A fast algorithm for the maximum clique problem, Discrete Applied Mathematics 120 (2002) 197–207.
[15] P.M. Pardalos, G.P. Rodger, A branch and bound algorithm for the maximum clique problem, Computers and Operations Research 19 (5) (1992) 363–375.
[16] T. Poranen, A simulated annealing algorithm for determining the thickness of a graph, Information Sciences 172 (2005) 155–172.
[17] L. Sánchez, I. Couso, J.A. Corrales, Combining GP operators with SA search to evolve fuzzy rule based classifiers, Information Sciences 136 (2001) 175–191.
[18] E. Tomita, T. Kameda, An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments, Journal of Global Optimization 37 (2007) 95–111.
[19] P.J.M. Van-Laarhoven, E. Aarts, Simulated Annealing: Theory and Applications, Kluwer, Dordrecht, 1987.
[20] X.S. Xu, J. Ma, An efficient simulated annealing algorithm for the minimum vertex cover problem, Neurocomputing 69 (2006) 913–916.
[21] X.D. Zhang, Z.L. Li, Graph Theory and Its Applications, Higher Education Publishing Company, Beijing, 2005.
[22] Q.F. Zhang, J.Y. Sun, E. Tsang, An evolutionary algorithm with guided mutation for the maximum clique problem, IEEE Transactions on Evolutionary Computation 9 (2) (2005) 192–200.
[23] J.Y. Zhang, J. Xu, Z. Bao, Algorithm for the maximum clique and independent set of graphs based on Hopfield networks, Journal of Electronics 18 (1996) 122–127.