Available online at www.sciencedirect.com
SCIENCE DIRECT
NEUROCOMPUTING
Neurocomputing 69 (2006) 913–916
www.elsevier.com/locate/neucom

Letter

An efficient simulated annealing algorithm for the minimum vertex cover problem

Xinshun Xu*, Jun Ma

School of Computer Science and Technology, Shandong University, Jinan, Shandong 250061, China

Received 7 November 2005; received in revised form 5 December 2005; accepted 5 December 2005
Available online 24 January 2006
Communicated by R.W. Newcomb

Abstract

The minimum vertex cover problem is a classic graph optimization problem. It is well known that it is an NP-complete problem. In this paper, an efficient simulated annealing algorithm is presented for the minimum vertex cover problem. In this algorithm, an acceptance function is defined for every vertex. This can help the algorithm in finding a near-optimal solution to a problem. Simulations are performed on several benchmark graphs, and the simulation results show that the proposed algorithm provides a high probability of finding optimal solutions.

Keywords: Vertex cover; NP-complete problem; Simulated annealing; Local minimum

1. Introduction

Given an undirected graph $G(V,E)$ with a vertex set V and an edge set E, the minimum vertex cover problem is to find a smallest subset $V' \subseteq V$ such that for each edge (a, b) in G $a \in V'$ or $b \in V'$ (or both), V' is said to be a vertex cover of G [5]. As an example, Fig. 1 shows a graph with 5 vertices and 6 edges, and the black vertex set {1,4} indicates a vertex cover (it is also a minimum cover in this graph). It is obvious than all the edges are covered by it.

The minimum vertex cover problem is a problem of central importance in computer science. It is very intractable [5]. In 1972, Karp has shown that the vertex cover problem is NP-complete [4]. However, because this problem has many important practical applications, especially in multiple sequence alignments for computational biochemistry [6], the problem is never abandoned. Many methods have been presented to solve the problem, e.g. the ratio-2 algorithm [2], evolutionary heuristic [5] and parameterized algorithm [1].

In 1985, Hopfield and Tank [3] proposed an approach of using a neural network to find a sub-optimal solution of the traveling salesman problem. This network is commonly referred as the Hopfield neural network. Since Hopfield and Tank's work, there has been growing interest in the Hopfield neural network because of its advantages over other approaches for solving combinatorial optimization problems. But the Hopfield neural network cannot usually get optimal or near-optimal solutions because the energy function comprises several terms, and there are many local minima [8].

In this paper, we propose an efficient simulated annealing algorithm for this problem, we call it ESA. A new acceptance function is introduced to help the ESA find an optimal solution to a given problem. The experimental results indicate that ESA has higher convergence rates to optimal solutions for benchmark graphs.

2. Problem formulation

For a given undirected graph $G = (V, E)$, V is vertex set and E is edge set. We let $|V| = n$, $|E| = m$. Binary variables d_{ij} $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, n)$ form the adjacency matrix of graph G. Each variable has only two values (1 or 0) which express the connection exists or not. In other words, if (i, j) is in E, then d_{ij} is 1 else d_{ij} is 0. At the same time, all

*Corresponding author. Tel./fax: +86 53188396694.
E-mail address: xuxinshun@sdu.edu.cn (X. Xu).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.12.016

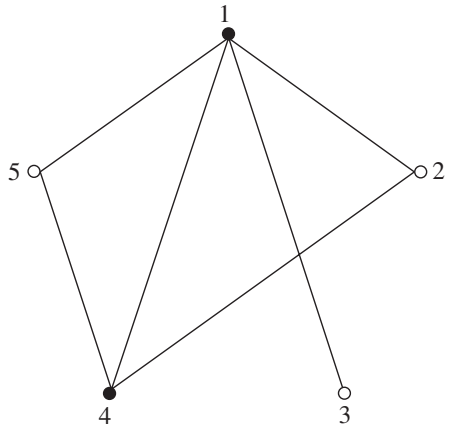

Fig. 1. A simple graph showing the minimum vertex cover problem.

graphs processed in this letter have no self-loops. That is to say, all variables d_{ii} $(i = 1, 2, \ldots, n)$ are equal to zero.

The state of the $\#i$ neuron can be determined by

$$v_i = \begin{cases} 1 & \text{if vertex } i \text{ is in the cover,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

So the number of vertices in the given graph determines the number of neurons. Thus the number of vertices in the cover can be expressed by

$$F_1 = \sum_{i=1}^{n} v_i \tag{2}$$

If an edge (i,j) is not covered by a cover, both v_i and v_j are zero. Then the constrained condition can be written as

$$F_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{d}_{ij} \overline{v_i \vee v_j}, \tag{3}$$

where $\vee$ is the logical OR, $\overline{X}$ means the complement of X.

The objective of the problem is to minimize F_1 with F_2 equal to zero. Thus, the objective function for the minimum vertex cover problem can be described as follows:

$$\begin{aligned} F &= A \sum_{i=1}^{n} v_i + B \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{d}_{ij} \overline{v_i \vee v_j} \\ &= A \sum_{i=1}^{n} v_i + B \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{d}_{ij} (v_i v_j - v_i - v_j) + B \sum_{i}^{n} \sum_{j}^{n} \mathrm{d}_{ij} \end{aligned} \tag{4}$$

where A,B are coefficients.

3. An efficient simulated annealing algorithm

The SA algorithm can be considered as a version of an "iterative improvement algorithm" which considers only specific transitions and terminates in the first local minima found [7]. Unlike this algorithm, simulated annealing allows various types of transitions in which some of them may be opposite towards achieving the goal. For instance, cost-increasing transitions are also accepted along with cost-decreasing transitions whereas an iterative improvement algorithm would allow only cost-decreasing ones to pass. However, it is proven that eventually simulated annealing produces a more optimal solution than the original iterative improvement algorithm [7]. But in most circumstances, the solution SA obtained is just a local optimal.

SA starts with an initial solution, s. A neighbor to this solution s', is then generated as the next solution by SA and the change in cost, $\Delta F(s, s')$ is evaluated. If a reduction in cost is found, the current solution is replaced by the generated neighbor, otherwise we decide with a certain probability whether s remains or s' becomes the current solution. The probability of accepting a transition that causes an increase, ΔF, in the cost is usually called the acceptance function as follows:

$$p = \mathrm{e}^{-\Delta F/T} \tag{5}$$

where T is the control parameter that corresponds to temperature in the analogy with the physical annealing process. In SA, the algorithm is started with a relatively high value of T, to have a better chance to avoid being prematurely trapped in a local minimum. The control parameter is lowered in steps until it approaches to zero. After termination, the final configuration is taken as the solution of the problem at hand.

In the ESA, s is the sequence of v_i $(i = 1, 2, \ldots, n)$, and s' is a neighbor of s if one v_i is changed, here v_i is selected randomly. For the minimum vertex cover problem it is well known that a vertex which has larger degree than the other vertices will be put into the cover with higher probability because such a vertex can cover more edges. Based on this, we modify the acceptance function as follows

$$p = \begin{cases} \mathrm{e}^{-\frac{\Delta F(1-Deg(v_i))}{T}} & v_i = 1, \\ \mathrm{e}^{-\frac{\Delta F(1+Deg(v_i))}{T}} & v_i = 0, \end{cases} \tag{6}$$

$$Deg(v_i) = \frac{Degree(i)}{EdgeNum}. \tag{7}$$

Where $Degree(i)$ is the degree of vertex i, its value is equal to the number of edges linking to the ith vertex, and $EdgeNum$ is a constant, equal to the total number of edges in a given graph.

Once $\Delta F > 0$ then we use Eq. (6) to determine whether a solution is replaced by its neighbor. By using Eq. (6), we have

1. if $v_i = 1$ in s', this means v_i's original value is 0. Once we accept s', the ith vertex is then selected into the cover set. From Eq. (6), p takes a larger value if the degree of the ith vertex is larger. This means s' will be accepted as a new solution with higher probability. That is to say, a vertex with larger degree will be selected into the cover set with higher probability. In contrast, if vertex i has a smaller degree, it will then be selected into the cover set with lower probability. For example, there are two vertices—the ith and the jth, with degree(i) > degree(j), and at the same T,

their reversions (from 0 to 1) make the object function change equally, i.e., $\Delta F(v_i) = \Delta F(v_j)$. Taking into account Eq. (6), we have

$$p(v_i) = e^{-\frac{\Delta F(v_i)(1-Deg(v_i))}{T}}, \quad (8)$$

$$p(v_j) = e^{-\frac{\Delta F(v_j)(1-Deg(v_j))}{T}}. \quad (9)$$

From Eq. (7) and the condition degree(i) > degree(j), we have

$$Deg(v_i) > Deg(v_j). \quad (10)$$

This means

$$\frac{\Delta F(v_i)(1-Deg(v_i))}{T} < \frac{\Delta F(v_j)(1-Deg(v_j))}{T}. \quad (11)$$

Then, we have the following relation between $p(v_i)$ and $p(v_j)$:

$$p(v_i) > p(v_j). \quad (12)$$

That is to say, the s' resulting from the reversion of v_i will be accepted as a new state with higher probability. On the contrary, $p(v_i)$ is equal to $p(v_j)$ with the acceptance function in the standard SA.

2. if $v_i = 0$, this means v_i's original value is 1. Once we accept s', the ith vertex is then removed from the cover set. From Eq. (6), p takes a smaller value if the degree of the ith vertex is larger. That means s' will be accepted as a new solution with lower probability if vertex has a larger degree. That is to say, a vertex with a larger degree will be removed from the cover set with lower probability. In contrast, if vertex i has a small degree, then it will be removed from the cover set with higher probability.

Table 1
Parameters used in the proposed algorithm

Parameters	Initial value	Meaning
A	1.0	The first coefficient in the object function
B	1.0	The second coefficient in the object function
T_0	50	Initial temperature
α	0.95	A constant to decrease temperature
L	100	# of steps to run without much change
ST	10	Maximum steps algorithm runs without much change

4. Simulation results

In order to assess the effectiveness of the proposed algorithm, extensive simulations were carried out on some benchmark graphs. These graphs are the complement graphs of some the DIMACS clique instances which can be downloaded from the URL http://dimacs.rutgers.edu/challenges/. The maximum clique problem is to find the maximum subset of vertices of graph G such that every two vertices are joined by an edge. It is very clear, if C is a clique in G, V–C is a vertex cover in G', where V is the vertex set of G, and G' is the complement of G. The parameters used in the proposed algorithm are shown in Table 1. The performance of the algorithm on the benchmarks is summarized in Table 2. Columns "Name", "n", and "$C(G)$" represent the name of the graph, the number of its vertices, and its maximum clique size, respectively that we are working with complements of these benchmark graphs, the maximum clique size is for the original instances. So in column "Cover", we give the minimum vertex cover size of the complement graphs which is equal to $n - C(G)$. For comparisons, the results of the original SA are also presented in column "SA". For each graph, two algorithms were run 100 times from different initial solution state. The optimal solution size and the success rate of finding an optimal solution are presented for each algorithm. The simulation results showed that the proposed algorithm can find optimal solutions to these benchmark graphs with a rate of near 100%.

5. Conclusions

We have proposed an efficient simulated annealing algorithm for the minimum vertex cover problem. A new acceptance function is used in the proposed algorithm.

Table 2
Simulation results on DIMACS benchmark graphs

Name	n	$C(G)$	Cover	SA		Proposed	
				Best	Rate	Best	Rate
MANN_a9	45	16	29	29	100%	29	100%
C-fat200-2	200	24	176	176	98%	176	100%
C-fat500-1	500	14	486	486	91%	486	98%
Hamming6-2	64	32	32	32	100%	32	100%
Johnson8-2-4	28	4	24	24	100%	24	100%
Johnson32-2-4	496	16	480	480	93%	480	99%
P_hat300-3	300	36	264	264	86%	264	98%
P_hat500-1	500	9	491	491	90%	491	99%
Sanr200_0.7	200	18	182	182	89%	182	98%

With this acceptance function, the proposed algorithm can find an optimal solution to a given graph with higher probability. The experimental results indicate that the algorithm has higher convergence rate, to optimal solutions on benchmark graphs, than the original SA.

Acknowledgments

The authors thank the anonymous reviewers for their helpful suggestions and comments. We also thank Professor Robert W. Newcomb for his suggestions and kind help. Finally, we acknowledge funding from SRF for ROCS, SEM.

References

[1] R.G. Downey, M.R. Fellows, Fixed parameter tractability and completeness II: completeness for W[1], Theor. Computer Sci. 141 (1–2) (1995) 109.
[2] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, 1979.
[3] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biol. Cybernetics 52 (4) (1985) 141.
[4] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computation, Plenum Press, New York, 1972, pp. 85–103.
[5] S. Khuri, T. Back, An evolutionary heuristic for the minimum vertex cover problem, in: G.I. Bonn (Ed.), Proceedings of KI-94 Workshops, the 18th German Annual Conference on Artificial Intelligence, 1994, pp. 83–84.
[6] R. Niedermeier, P. Rossmanith, Upper bounds for vertex cover further improved, in: Proceedings of STACS'99, the 16th Annual Symposium on Theoretical Aspects of Computer Science, Springer Lecture Notes in Computer Science, vol. 1563, Springer, 1999, pp. 561–570.
[7] D.T. Pham, D. Karaboga, Intelligent Optimisation Techniques: Genetic Algorithm, Tabu Search, Simulated Annealing, and Neural Networks, Springer, Berlin, 2000.
[8] K.A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research, Informs Journal on Computing 11 (1) (1999) 15–34.

Xinshun Xu received his B.S. degree from Shandong Normal University, Jinan, Shandong, China, in 1998, his M.S. degree from Shandong University (SDU), Jinan, Shandong, China, in 2002, and his Ph.D. Degree from Toyama University, Toyama, Japan, in 2005 and all in computer science. From 1998 to 2000, he was an Engineer in Shandong Provincial Education Department, Shandong, China. Between 2003 and 2005, he worked as a Research Assistant at Toyama University. He joined the School of Computer Science & Technology of SDU in 2005 as an Associate Professor. His main research interests include neural networks, machine learning, pattern recognition, text classification, bioinformatics, and optimization problems.

Jun Ma received his B.S., M.S. and Ph.D. from Shandong University of China, Ibaraki University of Japan and Kyushu University of Japan in 1982, 1988 and 1997, respectively, all in computer science. Now he is a professor at the School of Computer Science & Technology of Shandong University. His research interests include algorithms, AI, parallel computing, and web technology.