ELSEVIER

Information Processing Letters 53 (1995) 229–234

Information Processing Letters

A parallel algorithm for the minimum weighted vertex cover problem

Aristidis Likas, Andreas Stafylopatis *

Department of Electrical and Computer Engineering, National Technical University of Athens, 157 73 Zographou, Athens, Greece

Communicated by L. Kott; received 20 July 1992; revised 15 September 1994

Abstract

An original parallel algorithm is presented that stems from a suitable physical metaphor by considering each edge of the graph as a cell that is attracted by the corresponding vertices and finally moves towards the one of them that wins the competition.

Keywords: Heuristics; Optimization; Parallel algorithms; Vertex covering

1. Introduction

The minimum weighted vertex cover problem is defined as follows [2]:

Let $G = (V, E)$ be an undirected graph, where $V = \{1, \ldots, n\}$ represents the set of vertices and $E \subseteq \{(i, j) \mid i, j \in V\}$ represents the set of edges. Let also $m = |E|$ and $n = |V|$ the number of edges and vertices respectively and $A = (a_{ij})$ the vertex-edge incidence matrix of G ($i = 1, \ldots, n$, $j = 1, \ldots, m$), where $a_{ij} = 1$ if edge j is incident to vertex i, otherwise $a_{ij} = 0$. In addition there is a cost c_i associated with each vertex i ($i = 1, \ldots, n$). A minimum weighted vertex cover of G is a set $V' \subseteq V$ such that (i) for each edge $(i, j) \in E$, at least one of the vertices i and j belongs to V', i.e., V' *covers* E, and (ii) among all covers of E, V' has the smallest cost, i.e., $\sum_{i \in V'} c_i$ is minimum. The integer linear formulation of the problem can be stated as:

$$\begin{aligned}
&\text{minimize} && \sum_{i=1}^{n} c_i x_i, \\
&\text{subject to} && \sum_{i=1}^{n} a_{ij} x_i \geqslant 1 \ (j = 1, \ldots, m) \\
& && \text{and } x_i \in \{0, 1\} \ (i = 1, \ldots, n).
\end{aligned}$$

Finding a minimum weighted vertex cover of an arbitrary graph G is an NP-complete problem [2,5]. For this reason, heuristics have been devised that require polynomial time but lead to suboptimal solutions. A sequential "greedy" algorithm of this kind is presented in [5]. Also connectionist approaches have been developed that are based on the *Boltzmann Machine method* [10] and on the *Competitive Activation method* [1,7–9].

* Corresponding author.

In the first approach, the solution is obtained indirectly as complementary to the solution of the maximum independent set problem that is actually solved. The main disadvantage of the Boltzmann Machine approach is that the update of each unit in the network is performed sequentially and thus the method cannot be efficiently parallelized. Moreover, as stated in [10], the particular characteristics of this problem make more convenient a connectionist approach based on the competitive activation mechanism. The latter method has also the advantage that it can be expressed in terms of a system of differential equations and can be parallelized efficiently.

In the competitive activation approach, each vertex of G corresponds to a processing element (node) of the network and each edge to a communication channel between two nodes. Initially, the activation level a_i of each node i is set to zero. The evolution of the network respects the following dynamics for each node i:

$$\frac{da_i}{dt} = \left(in_i - \frac{A}{c_i} a_i \right) (1 - a_i),$$

where $in_i = \Sigma_j out_{ij}$ and $A > 0$. The output from node j to node i is $out_{ij} = (1 + a_i A / c_i)(1 - a_j)$. It can be shown that, under the above dynamics, the network converges to an equilibrium state with either $a_i = 1$ or $a_i = 0$ for each node i. If $a_i = 1$ then node i is incorporated in the solution set. The equilibrium state has the property that, if there is a connection between nodes i and j, then these nodes cannot be both deactivated. This means that equilibrium states correspond to feasible solutions of the vertex covering problem.

The algorithm proposed in this paper deals directly with the vertex cover problem and is also expressed in terms of a system of differential equations that can be solved iteratively and in parallel. We prove that the iterative procedure reaches an equilibrium state which indicates a feasible solution to the problem. In this approach, the vertices of a given graph compete for the allocation (covering) of the resources (edges) associated with them, and this competition can be simulated in terms of a physical metaphor that is described in the next section.

2. The parallel algorithm

2.1. A natural metaphor

The proposed algorithm is based on the following natural metaphor. Consider that each vertex i of the graph corresponds to a *still* body i and each undirected edge (i, j) corresponds to a cell $p(i, j)$. Since every edge is undirected, the symbols $p(i, j)$ and $p(j, i)$ refer to the same cell. There is a kind of *antagonistic attraction* over each cell $p(i, j)$ which accepts opposite forces from the two bodies i and j, each of which tries to attract the cell towards its own site. The cell is moving in the direction of the greater force. The strength of the force imposed by a body i upon its associated cells $p(i, j)$ is determined by the *potential* of the body, which depends upon the cost of the corresponding vertex in the graph and on the positions of the cells $p(i, j)$. In case a cell $p(i, j)$ is moving towards body i (i.e., the potential of body i is greater than the potential of body j), then the potential of i is further increased while the potential of j is further decreased. This is reasonable since, once body i seems to win the competition over body j for the cell $p(i, j)$, its potential should be increased in order to be able to compete for the other associated cells $p(i, k)$ with increased probability of success. On the other hand, the reduction in the potential of body j can be justified by the fact that a vertex that is losing a competition for a cell should be discouraged from allocating other cells, since this may lead to inefficient solutions from the minimality point of view. In this way, the competition among bodies for the allocation of cells is carried out indirectly through the positions of the cells relatively to the corresponding body sites.

The cells are initially located at positions of equal finite distance from the associated bodies. Due to the imposed forces, each cell begins to move and, at the end, every cell will have arrived at the one of the two bodies having the greater final potential. In case a body has obtained *at least one cell*, the corresponding vertex should be included in the solution set of the vertex covering problem, since it has the responsibility for covering at least one edge. Otherwise, the vertex is considered as a loser of the competition and is not incorporated in the solution set of the problem.

2.2. Formulation of the algorithm

To each edge (i, j) of the undirected graph, we associate a cell $p(i, j)$. Thus, the number of cells is equal to the number of existing undirected edges. Such a cell can move along the line determined by the coordinates of vertices i and j. The position of the cell $p(i, j)$ along this line is expressed in terms of the quantities r_{ij} and r_{ji}, which are specified as follows. If a cell $p(i, j)$ is positioned in the middle of the line segment connecting vertex i to vertex j, then $r_{ij} = 0$ and $r_{ji} = 0$. In the case where the cell $p(i, j)$ is closer to vertex i then $r_{ij} > 0$ and $r_{ji} < 0$. The opposite holds in the case where the cell $p(i, j)$ is closer to vertex j. With respect to cell $p(i, j)$, vertex i is considered as the point with $r_{ij} = r_{max}$ and $r_{ji} = -r_{max}$, where $r_{\max} > 0$. Similarly, vertex j is considered as the point with $r_{ji} = r_{max}$ and $r_{ij} = -r_{max}$. A constraint imposed by the algorithm is that at every time instant $r_{ij} = -r_{ji}$. Fig. 1(a) presents an initial system configuration with each cell $p(i, j)$ positioned at $r_{ij} = r_{ji} = 0$. Fig. 1(b) displays a possible final system configuration, where the marked vertices are those that belong to the solution set (i.e., those that have been assigned at least one cell).

The potential of vertex i $(i = 1, \dots, n)$ at each time instant t is determined as follows,

$$U_i(t) = \sum_k r_{ik}(t) - c_i, \tag{1}$$

where the summation concerns all cells $p(i, k)$ that are associated with vertex i. The *motion equation* of each cell $p(i, j)$ is

$$\frac{du_{ij}(t)}{dt} = \frac{U_i(t)}{S_i} - \frac{U_j(t)}{S_j} \tag{2}$$

and

$$r_{ij}(t) = r_{max} f\big(u_{ij}(t)\big). \tag{3}$$

In these equations we denote by S_i $(i = 1, \dots, n)$ the number of cells corresponding to vertex i, or, equivalently, the degree of vertex i. Also, we denote by f the sigmoid function

$$f(x) = \tanh(\alpha x) \tag{4}$$

which constrains its values to the interval $[-1, 1]$. The constant $\alpha > 0$ is a parameter that serves to adjust the slope of the sigmoid.

From Eq. (2) it is apparent that $du_{ij}/dt = -du_{ji}/dt$. Thus, based on the fact that f is odd, we come to the conclusion that, if initially $u_{ij}(0) = -u_{ji}(0)$, then at each time instant we will have $u_{ij}(t) = -u_{ji}(t)$ and $r_{ij}(t) = -r_{ji}(t)$. Moreover, it is clear that the larger the values of r_{ij} and the smaller the value of c_i, the greater the probability that du_{ij}/dt is positive, i.e., that cell $p(i, j)$ is moving towards vertex i. Eqs. (1)–(3) describe the

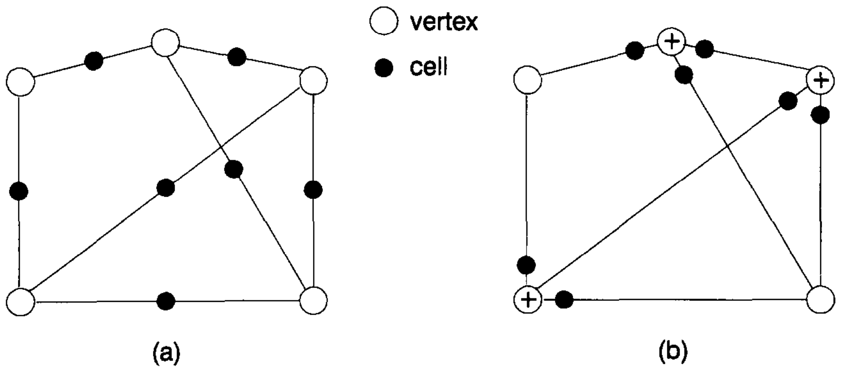

Fig. 1. (a) An initial system state. (b) A final system state and the corresponding solution set.

basic relations governing the evolution of the system (the quantities u_{ij} and r_{ij}) over time.

The specification of the parallel algorithm is based on Eqs. (1)–(3) and can be stated as follows:

1. Set $t = 0$, and initialize to small near-zero values: $u_{ij}(0) = -u_{ji}(0) = \delta$ and $r_{ij}(0) = f(u_{ij}(0))$, where for each cell the value δ is randomly chosen in the interval $(-10^{-4}, 10^{-4})$.
2. For each time instant t and for all cells $p(i, j)$, use Eqs. (1) and (2) to compute $\Delta u_{ij}(t)$.
3. For all cells $p(i, j)$ compute $u_{ij}(t + \Delta t)$ on the basis of the first-order Euler approximation:
$$u_{ij}(t+\Delta t) = u_{ij}(t) + \Delta u_{ij}(t)\Delta t. \tag{5}$$
Then set $u_{ji}(t + \Delta t) = -u_{ij}(t + \Delta t)$.
4. For all cells $p(i, j)$ compute $r_{ij}(t + \Delta t)$:
$$r_{ij}(t+\Delta t) = \tanh\big(\alpha u_{ij}(t+\Delta t)\big). \tag{6}$$
Also, set $r_{ji}(t + \Delta t) = -r_{ij}(t + \Delta t)$.
5. Increment t by Δt. Terminate iterative procedure (go to step 6) if $|r_{ij}(t + \Delta t) - r_{ij}(t)| < \varepsilon$ for all cells $p(i, j)$, where ε is a very small positive constant, otherwise go to step 2.
6. For each cell $p(i, j)$, such that $r_{ij} > 0$, incorporate vertex i to the final solution set V'.

In practice, upon termination of the algorithm, we have that $|r_{ij}|$ is essentially equal to r_{max} for all cells. As a matter of fact, if the slope of the sigmoid is sufficiently large, the system is expected to reach an equilibrium state in which all cells will have moved to one of the ends of the corresponding edge.

2.3. Proof of convergence

In order to prove that the above described dynamical system converges to an equilibrium state, it is sufficient to show that there exists a quantity $E(t)$ (called *energy* in the context of neural networks [3,4]), which constitutes a Liapunov function for the given system, i.e., has the property that $dE/dt \leqslant 0$.

Proposition 1. *The function*

$$E(t) = -\tfrac{1}{4}\sum_i \sum_j r_{ij}(t)\left(\frac{1}{S_i}\sum_k r_{ik}(t) - \frac{1}{S_j}\sum_l r_{jl}(t)\right)$$
$$-\tfrac{1}{2}\sum_i \sum_j r_{ij}(t)\left(\frac{c_j}{S_j} - \frac{c_i}{S_i}\right) \tag{7}$$

constitutes a Liapunov function of the dynamical system described by Eqs. (1)–(3).

Proof. The following holds,

$$\frac{dE}{dt} = -\sum_i \sum_j \frac{\vartheta E}{\vartheta r_{ij}} \frac{dr_{ij}}{du_{ij}} \frac{du_{ij}}{dt}, \tag{8}$$

where $dr_{ij}/du_{ij} = d\tanh(\alpha u_{ij})/du_{ij} = \alpha(1 - (\tanh(\alpha u_{ij}))^2)$. It is obvious that $dr_{ij}/du_{ij} > 0$ and $dr_{ij}/du_{ij} = dr_{ji}/du_{ji}$.

Taking into account that $r_{ij}(t) = -r_{ji}(t)$, it can be verified using Eqs. (1), (2) and (7) that

$$\frac{du_{ij}}{dt} = \frac{\vartheta E}{\vartheta r_{ij}} - \frac{\vartheta E}{\vartheta r_{ji}}. \tag{9}$$

Using this result Eq. (8) takes the form

$$\frac{dE}{dt} = -\sum_i \sum_{j,i<j} \frac{dr_{ij}}{du_{ij}}\left(\frac{\vartheta E}{\vartheta r_{ij}}\frac{\delta r_{ij}}{\delta t} + \frac{\vartheta E}{\vartheta r_{ji}}\frac{dr_{ji}}{dt}\right) \tag{10}$$

$$= -\sum_i \sum_{j,i<j} \frac{dr_{ij}}{du_{ij}} \frac{\vartheta E}{\vartheta r_{ij}}\left(\frac{\vartheta E}{\vartheta r_{ij}} - \frac{\vartheta E}{\vartheta r_{ji}}\right)$$
$$-\frac{\vartheta E}{\vartheta r_{ji}}\left(\frac{\vartheta E}{\vartheta r_{ji}} - \frac{\vartheta E}{\vartheta r_{ji}}\right) \tag{11}$$

and finally

$$\frac{dE}{dt} = -\sum_i \sum_{j,i<j} \frac{dr_{ij}}{du_{ij}}\left[\left(\frac{\vartheta E}{\vartheta r_{ij}}\right)^2 + \left(\frac{\vartheta E}{\vartheta r_{ji}}\right)^2\right] \leqslant 0 \tag{12}$$

since $dr_{ij}/du_{ij} > 0$ as stated previously. □

3. Experimental results and performance considerations

Comparative experiments have been performed in order to examine the performance of the algorithm against the competitive activation approach that is considered very effective for this particular problem (especially in the cardinality case) and outperforms the "greedy" sequential heuristic.

In particular, we used randomly constructed graphs of sizes varying between 20 and 80 vertices. For the construction of each graph, edge (i,j) was included in the graph with probability $p = 0.1$. (This rather small probability value was selected so that relative sparse graphs were constructed.) If, at the end of the construction procedure, a vertex with no connections were found, an edge was added in the graph connecting this particular vertex with another randomly selected vertex. For each graph size three cases were considered that are characterized by increasing difficulty: (a) cardinality problems (where the cost of each vertex is equal to 1), (b) "regular" cost problems, where the cost of each vertex was an integer value randomly chosen in the range from 1 to 4, and (c) "irregular" cost problems, where half of the vertices were assigned cost values randomly chosen in the range from 1 to 4, while the cost values assigned to the remaining vertices were randomly chosen in the range from 1 to 40.

For each graph size, 20 instances were randomly constructed and tested. To each constructed graph, both the method developed here and the competitive activation method were applied and the costs of the resulting covers were compared. Moreover, for graph instances of size up to 40 the optimal solution was found using exhaustive search. (For larger graphs the computational cost of exhaustive search is excessively high.)

Table 1 summarizes the main results obtained from the three kinds of experiments with P and C denoting the cost of the parallel algorithm and of the competitive activation approach respectively. For each type and size of problem, the entries of the table indicate the percentage of instances for which the condition of the corresponding column holds.

The main conclusions that can be drawn from the experiments are the following. In the case of the cardinality problem, the methods exhibit equivalent performance, yielding near-optimal vertex covers, with the competitive activation method sometimes resulting in covers of slightly lower cost. It must be noted that, in all tested cases, the corresponding solutions either were equal or differred by only one unit.

As far as graphs with regular cost are concerned, the displayed results make clear the superiority of our method in most of the cases. This superiority becomes total in the case of difficult problems with irregular cost values, where our method provided a better solution in almost all of the tested cases with a significantly lower cost.

Another metric of the quality of a solution obtained for a given problem instance I is the *approximation ratio* which is defined as $h(I)/opt(I)$, where $h(I)$ denotes the cost of the obtained solution and $opt(I)$ denotes the cost of the optimal solution [6]. Table 2 displays the average approximation ratio achieved by the par-

Table 1
Comparative results

Size	Cardinality			Regular cost			Irregular cost		
	$P<C$	$P=C$	$P>C$	$P<C$	$P=C$	$P>C$	$P<C$	$P=C$	$P>C$
20	7%	86%	7%	60%	13%	27%	87%	0%	13%
30	26%	60%	14%	54%	13%	33%	93%	6%	0%
40	13%	74%	13%	33%	13%	54%	87%	0%	13%
50	20%	47%	33%	60%	0%	40%	87%	0%	13%
60	20%	50%	30%	60%	20%	20%	100%	0%	0%
80	25%	40%	35%	75%	25%	0%	100%	0%	0%

Table 2
Quality of obtained solutions

Size	Approximation ratio		
	Cardinality	Regular cost	Irregular cost
20	1.01	1.09	1.17
30	1.06	1.12	1.19
40	1.06	1.14	1.22

allel algorithm for problems of moderate size. The results suggest that the method can provide solutions of high quality.

An additional important feature is that our algorithm does not require the adjustment of any parameters in order to work properly. This is an advantage over all known connectionist approaches to combinatorial optimization that require, in general, experimentation and tuning of some critical parameters in order to be effective. On the other hand, it must be noted that the competitive activation algorithm is characterized by better behaviour in terms of execution time. This is due to the fact that, in the connectionist case, complexity is determined by the number of graph vertices, while in our case complexity depends on the number of edges.

The algorithms were implemented and tested on a sequential computer and, specifically, on a SPARC workstation. An implementation aiming to reveal the full potential of our approach (in terms of execution speed), should exploit the inherent parallelism of the algorithm. A parallel implementation on a transputer-based parallel machine and the examination of its behaviour on large-scale problems (graphs containing thousands of vertices) constitute an objective of future research.

Another objective is related to the application of the antagonistic attraction mechanism introduced here to other covering and packing problems such as the set covering, set partitioning and set packing problems [2,6]. Such a task is not straightforward and an adaptation of the algorithm is required in order to deal effectively with the particular characteristics of each problem.

References

[1] P. Bourret, G. Reggia and M. Samuelides, Réseaux neuronaux, Teknea, Toulouse, 1991.

[2] R. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness* (W.H. Freeman and Company, San Francisco, 1979).

[3] J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Nat. Acad. Sci. USA* **79** (1982) 2554-2558.

[4] J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems, *Biological Cybernetics* **52** (1985) 141–152.

[5] D. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. System Sci.* **9** (1974) 256–278.

[6] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[7] Y. Peng, A connectionist approach to vertex covering problems, in: *Proc. IJCNN*, Washington, 1989.

[8] Y. Peng, J.A. Reggia and T. Li, A connectionist approach to vertex covering problems, *Internat. J. Neural Systems* **3** (1) (1992).

[9] J. Reggia, Y. Peng and P. Bourret, Recent applications of competitive activation mechanisms, in: E. Gelenbe, ed., *Neural Networks: Advances and Applications* (North-Holland, Amsterdam, 1991) 31–62.

[10] V. Zissimopoulos, V. Paschos and F. Pekergin, On the approximation of NP-complete problems by using the Boltzmann Machine method. The cases of some covering and packing problems, *IEEE Trans. Comput.* **40** (12) (1991) 1413–1419.