A Linear-Time and Linear-Space Algorithm for the Minimum Vertex Cover Problem on Giant Graphs*

Hong Xu, T. K. Satish Kumar, Sven Koenig
University of Southern California, Los Angeles, California 90089, the United States of America
hongx@usc.edu tkskwork@gmail.com skoenig@usc.edu

Abstract

In this paper, we develop the *message passing based linear-time and linear-space MVC algorithm* (MVC-MPL) for solving the minimum vertex cover (MVC) problem. MVC-MPL is based on heuristics derived from a theoretical analysis of message passing algorithms in the context of belief propagation. We show that MVC-MPL produces smaller vertex covers than other linear-time and linear-space algorithms.

Introduction

Given an undirected graph $G = \langle V, E \rangle$, a *vertex cover* (VC) of G is defined as a set of vertices $S \subseteq V$ such that every edge in E has at least one of its endpoint vertices in S. A *minimum vertex cover* (MVC) of G is a vertex cover of minimum cardinality. The MVC problem is to find an MVC for a given graph. It is an NP-hard problem (Karp 1972) that has been used across a wide range of application domains, such as crew scheduling, VLSI design, nurse rostering and industrial machine assignments (Cai et al. 2013). Many heuristics have been developed to tackle the MVC problem and its generalizations (Cai et al. 2013; Xu, Kumar, and Koenig 2016). However, none of these algorithms are linear-time and linear-space in the number of vertices and edges. (Henceforth, we use the term "linear" to refer to being linear-time and linear-space in the number of vertices and edges.) Therefore, they are difficult to apply to giant graphs with billions of vertices and edges.

Belief propagation (BP) is a well-known technique used for solving queries such as probability marginalization and maximum-a-posteriori estimation in probabilistic models (Yedidia, Freeman, and Weiss 2003). *Message passing* is a class of techniques that generalize BP. It, too, has been applied to problems across various fields, including the Ising model in statistical physics and error correcting codes in information science (Yedidia, Freeman, and Weiss 2003).

In this paper, we develop the *message passing based linear MVC algorithm* (MVC-MPL) that solves the problem heuristically using the theory of message passing. Empirically, we show that MVC-MPL produces smaller VCs than other linear MVC algorithms.

*The research at the University of Southern California was supported by NSF under grant numbers 1409987 and 1319966.

Background

A Linear Factor-2 Approximation Algorithm

A well-known linear factor-2 approximation algorithm (MVC-2) for solving the MVC problem works as follows (Vazirani 2003). In each iteration, the algorithm arbitrarily selects an uncovered edge, then marks it as well as the edges incident on its two endpoint vertices as being covered, and adds its endpoint vertices into the vertex cover VC. The algorithm then proceeds to the next iteration until all edges are marked as being covered. This is the only existing well-known linear MVC algorithm that is known to the authors.

The Warning Propagation Algorithm

The *warning propagation algorithm* is a special message passing algorithm where messages can only take one of two values, namely 0 or 1 (Weigt and Zhou 2006). (Weigt and Zhou 2006) developed an algorithm that uses warning propagation to solve the MVC problem and analyzed it theoretically. In their algorithm, messages are passed between adjacent vertices. A message of 1 from $v_i \in V$ to $v_j \in V$ indicates that v_i is not in the MVC and thus it "warns" v_j to be included in the MVC. Their theoretical analysis mainly focused on Erdős-Rényi (ER) random graphs, in which each edge is generated with a constant probability (Erdős and Rényi 1959). They show that, on an infinitely large ER random graph, a message has a probability of $W(c)/c$ to be equal to 1, where c is the average degree of vertices and W is the Lambert-W function.

The Algorithm

MVC-MPL treats the input graph as if it were generated by the ER model. It repeatedly selects an arbitrary vertex v and decides whether it should be included in or excluded from the VC. MVC-MPL does this by considering those vertices adjacent to v which have not yet been included in the VC. (No vertex adjacent to v has already been excluded from the VC since otherwise v must necessarily be included in the VC.) We use $k(v)$ to denote the number of such adjacent vertices. Each of these vertices sends a message of 1 to v with probability $W(c)/c$ under the warning propagation algorithm. Vertex v must be included in the VC iff it receives at least one message of 1. Assuming independence, the probability that v is excluded from the VC is thus $(1 - W(c)/c)^{k(v)}$. Consequently, MVC-MPL excludes v from the VC with this

Input Graph			Solution Size			Running Time (milliseconds)		
Instance	$\|V\|$	$\|E\|$	MVC-MPL	MVC-L	MVC-2	MVC-MPL	MVC-L	MVC-2
bn-human-BNU-1-0025864-session-1-bg	696,338	143,158,340	647,568	659,013	686,776	724	925	1,101
bn-human-BNU-1-0025864-session-2-bg	692,957	133,727,517	644,157	655,414	683,248	702	882	1,016
soc-livejournal	4,033,137	27,933,063	2,148,197	2,205,385	2,591,926	893	971	731
soc-ljournal-2008	5,363,201	79,023,143	3,127,083	3,623,388	4,908,058	1,200	1,363	1,492
soc-livejournal07	5,204,176	49,174,621	2,882,334	2,913,930	3,522,680	1,390	1,610	1,194
tech-as-skitter*	1,696,415	11,095,299	624,654	695,988	891,280	253	252	193
tech-ip	2,250,498	21,644,715	69,525	122,870	132,640	681	497	176
web-baidu-baike	2,141,330	17,794,839	745,685	784,284	1,063,178	527	528	300
web-hudong	1,984,484	14,869,484	713,449	743,685	1,061,712	406	390	248
web-wiki-ch-internal	1,930,275	9,359,108	323,142	351,770	418,946	336	309	130

* http://www.caida.org/tools/measurement/skitter/

Table 1: The "Instance" column shows the names of the input graphs; the "$|V|$" and "$|E|$" columns show the numbers of vertices and edges in the graphs, respectively. The next 6 columns show the size of the VC and running time of each algorithm on each instance, respectively. For each instance, the size of the VC and running time are averages over 20 repeated runs.

probability and includes it otherwise. If v is excluded, MVC-MPL marks all its adjacent vertices as being included in VC. Algorithm 1 shows the details of MVC-MPL, where ∂v is the set of adjacent vertices of v and IS stands for the independent set. Its time and space complexities are both $\mathcal{O}(|V| + |E|)$.

Experimental Evaluation

We add another algorithm MVC-L for comparison purposes. MVC-L works similar to MVC-MPL, except that it excludes a vertex v with probability $1/(k(v)+1)$ instead of $p_0^{k(v)}$. In other words, $p_0^{k(v)}$ in line 8 of Algorithm 1 is replaced by $1/(k(v)+1)$. The intuition is to include high-degree vertices in the VC with higher probabilities.

In our experiments, since the average degrees of vertices in all input graphs are larger than e, we approximated the Lambert-W function using the first 3 terms of Equation 4.19 in (Corless et al. 1996), i.e., $W(c) = L_1 - L_2 + L_2/L_1 + \mathcal{O}\left((L_2/L_1)^2\right)$, where $L_1 \equiv \log c$ and $L_2 \equiv \log L_1$.

All algorithms were implemented in C++, compiled by GCC 6.3.0 with the "-O3" option and run on a GNU/Linux workstation with Intel Xeon Processor E3-1240 v3 (8MB Cache, 3.4GHz) and 16GB RAM. We fixed the vertex or edge visitation orders to those given in the input files.

All input graphs are giant graphs selected from the Brain Networks, Social Networks, Technological Networks and Web Graphs categories in the Network Repository (http://networkrepository.com) (Rossi and Ahmed 2015; Mislove et al. 2007; Niu et al. 2011). Table 1 shows our experimental results. We see that all algorithms terminated quickly and MVC-MPL produced significantly smaller VCs than MVC-L and MVC-2.

Algorithm 1: MVC-MPL

```
1  Function MVC-MPL(G = ⟨V, E⟩)
       Input: G: The graph to find an MVC for.
       Output: A VC of G.
2      Initialize VC = ∅ and IS = ∅;
3      c := average degree of vertices in G;
4      p_0 := 1 − W(c)/c;
5      while ∃v ∈ V \ (VC ∪ IS) do
6          k(v) := |{u ∈ ∂v | u ∉ VC}|;
7          Draw a random real number r uniformly at random from [0, 1];
8          if r < p_0^{k(v)} then
9              Add v to IS;
10             Add all u ∈ ∂v to VC;
11         else
12             Add v to VC;
13     return VC;
```

Conclusion

We developed MVC-MPL, a new linear algorithm for solving the MVC problem on giant graphs. Empirically, we showed that MVC-MPL terminated fast and produced smaller VCs than MVC-L and MVC-2.

References

Cai, S.; Su, K.; Luo, C.; and Sattar, A. 2013. NuMVC: An efficient local search algorithm for minimum vertex cover. *Journal of Artificial Intelligence Research* 46(1):687–716.

Corless, R. M.; Gonnet, G. H.; Hare, D. E. G.; Jeffrey, D. J.; and Knuth, D. E. 1996. On the LambertW function. *Advances in Computational Mathematics* 5(1):329–359.

Erdős, P., and Rényi, A. 1959. On random graphs I. *Publicationes Mathematicae* 6:290–297.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, New York. 85–103.

Mislove, A.; Marcon, M.; Gummadi, K. P.; Druschel, P.; and Bhattacharjee, B. 2007. Measurement and analysis of online social networks. In *ACM Internet Measurement Conference*, 29–42.

Niu, X.; Sun, X.; Wang, H.; Rong, S.; Qi, G.; and Yu, Y. 2011. Zhishi.me - weaving Chinese linking open data. In *the International Semantic Web Conference*, 205–220.

Rossi, R. A., and Ahmed, N. K. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI Conference on Artificial Intelligence*, 4292–4293.

Vazirani, V. V. 2003. *Approximation Algorithms*. Springer.

Weigt, M., and Zhou, H. 2006. Message passing for vertex covers. *Physical Review E* 74(4):046110.

Xu, H.; Kumar, T. K. S.; and Koenig, S. 2016. A new solver for the minimum weighted vertex cover problem. In *the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*, 392–405.

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2003. Understanding belief propagation and its generalizations. *Exploring Artificial Intelligence in the New Millennium* 8:239–269.