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

Hong Xu T. K. Satish Kumar Sven Koenig

hongx@usc.edu tkskwork@gmail.com skoenig@usc.edu

University of Southern California, Los Angeles, California 90089, the United States of America

The 10th International Symposium of Combinatorial Search

I. 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.

II. Algorithms in the Comparison

- The MVC problem: The problem to find a vertex cover of minimum cardinality on a given undirected graph.
- MVC-2: A factor-2 approximation algorithm. It arbitrarily selects an edge and adds both endpoint vertices into the vertex cover. It repeats this procedure until all edges are covered.
- MVC-L: An algorithm based on the intuition that vertices with higher degrees are more likely to be in the vertex cover.
- MVC-MPL: Our algorithm inspired by warning propagation.

III. Warning Propagation for the MVC Problem

Proposed and analyzed by Weigt and Zhou (2006).
There is a message of either 0 or 1 along each direction of each edge. Assuming u and v are two adjacent vertices. u sends v a message of 1 to "warn" v to indicate that u will not be selected in the vertex cover. Otherwise, u sends v a message of 0.

- u sends a message of 0 to v iff u only receives messages of 0 from its adjacent vertices other than v.
- A vertex is in the vertex cover iff it has at least one incoming message of 1.
- Assuming independence of the probabilities of messages, a given vertex can be chosen to be in or not in the vertex cover based on the probability of having at least one incoming message of 1.

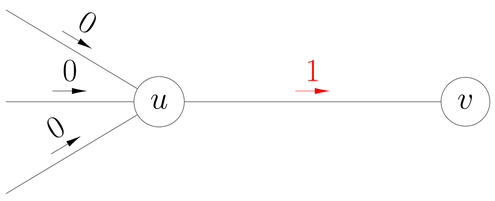

Fig. 1: u sends a message of 1 to v since all other incoming messages are 0.

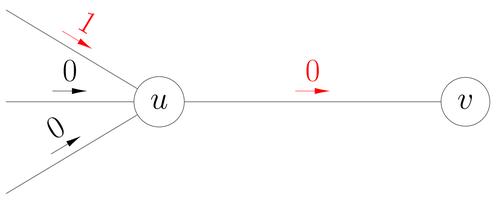

Fig. 2: u sends a message of 0 to v since one of its incoming messages is 1.

IV. MVC-MPL

Algorithm 1: MVC-MPL

Function `MVC-MPL`($G = \langle V, E \rangle$)
  Input: G: The graph to find an MVC for.
  Output: A VC of G.
  Initialize $VC = \emptyset$ and $IS = \emptyset$;
  $c :=$ average degree of vertices in G;
  $p_0 := 1 - W(c)/c$; `// Fraction of zero messages`
  while $\exists v \in V \setminus (VC \cup IS)$ **do**
    $k(v) := |\{u \in \partial v \mid u \notin VC\}|$; `// Number of adjacent vertices`
    Draw a random real number r uniformly at random from $[0, 1]$;
    if $r < p_0^{k(v)}$ **then**
      Add v to IS;
      Add all $u \in \partial v$ to VC;
    else
      Add v to VC;
  return VC;

VC: vertex cover IS: independent set
∂v: set of adjacent vertices of v $W(\cdot)$: Lambert-W function

V. Experimental Results

Input Graph			Vertex Cover 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

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. All instances are taken from the Network Repository (Rossi and Ahmed 2015) of various categories.

- MVC-MPL won on almost all instances in terms of vertex cover sizes.
- MVC-MPL won on 3 out of 10 instances in terms of running times. For those instances on which it was slower, it still terminated fast.

VI. References

Rossi, Ryan A. and Nesreen K. Ahmed (2015). "The Network Data Repository with Interactive Graph Analytics and Visualization". In: *AAAI Conference on Artificial Intelligence*, pp. 4292–4293. URL: `http://networkrepository.com`.
Weigt, Martin and Haijun Zhou (2006). "Message passing for vertex covers". In: *Physical Review E* 74.4, p. 046110. DOI: `10.1103/PhysRevE.74.046110`.

Acknowledgments

The research at the University of Southern California was supported by the National Science Foundation under grant numbers 1409987 and 1319966.