Comparison of Ordering Heuristics of Depth-First Search for Minimum Connected Vertex Cover

Tahmid A. Khan
Eletrical & Computer Engineering
North South University
Dhaka, Bangladesh
tahmid.khan1@northsouth.edu

Ahsanur Rahman
Eletrical & Computer Engineering
North South University
Dhaka, Bangladesh
ahsanur.rahman@northsouth.edu

***Abstract*—The Minimum Connected Vertex Cover (CVC) problem, a variant of the Minimum Vertex Cover (VC) problem, aims to find a minimum-cardinality subset of the vertices of a given graph such that the subgraph induced by those vertices is connected. A fundamental result related to the CVC problem is that if the leaves of a spanning tree of the graph are independent (*i.e.,* not adjacent to each other in the graph), then the internal vertices of this tree provides a 2-approximation for the graph's CVC. In this paper, building upon the fact that the depth-first search algorithm (DFS) can be used to obtain an independent-leaves spanning tree of a graph, we compare the efficacy of variations of DFS for obtaining an approximation of the optimal CVC. In particular, we compare four different ordering heuristics commonly used in greedy graph coloring (with two variations, forward and reverse, for each heuristic), to use with DFS: decreasing-degree, smallest-degree-last, smallest-log-degree-last, and saturation-degree. Our experiments on small real-world graphs show that, at least for graphs with small number of vertices, the decreasing-degree heuristic almost always performs at least as good as the others in terms of approximation ratio, and the forward variant of each ordering almost always performs at least as good as the reverse variant.**

I. Introduction

Given an undirected connected graph $G = (V, E)$, a vertex subset $V' \subseteq V$ is called a *vertex cover* of G if every edge $(u, v) \in E$ is incident to some vertex in V', that is, $u \in V'$ or $v \in V'$. The *minimum vertex cover* problem (VC) aims to find a vertex cover with minimum cardinality. The *minimum connected vertex cover* problem (CVC) is a variation of the vertex cover problem where the aim is to find a minimum vertex cover C of a graph G such that the subgraph of G induced by the vertices C is connected.

First introduced by Garey and Johnson [1], the CVC problem typically arises in situations that necessitate finding a vertex cover of a network while maintaining certain connectivity constraints on possible solutions. As an example, canonical use case of CVC, from the field of wireless network design, is the problem of minimizing the number of relay stations that need to be placed on a network's nodes so that they form a connected subnetwork and every transmission link is incident to at least one relay station [2].

Contributions: Our main contribution in this paper is a comparative study of the efficay four different vertex ordering heuristics commonly used in greedy graph coloring algorithms, for obtaining a CVC using depth-first search. The source code used to run the experiments, along with instructions on how to compile and run them, can be found at the following remote Git repository: `https://github.com/tahmid-khan/cvc-approximation`

II. Related Work

Savage [3] proved that the internal vertices of a DFS spanning tree give a 2-approximation for the CVC; this is one of the main results our approach is based on. Most of the approaches to approximating CVC make use of approximations of *tree cover* problem, in which the objective is to compute a minimum edge set T such that T is connected and every edge outside of T is adjacent to at least one in T [4]. Fujito *et al*. [4] proposed a $\mathcal{O}(\log^2 n)$-time algorithm that uses maximal matchings to find a 2-approximation to CVC. Better approximation algorithms for CVC exist for special connected graphs, such as a $\frac{2k}{k+2} + \mathcal{O}(1/n)$-approximation algorithm for k-regular graphs [5].

III. Methodology

Our methodology is based on the following facts: (1) the internal vertices of an independence tree form a connected vertex cover, and (2) the d-leaves (leaves that are not the root) of a DFS spanning tree T of a graph G are G-independent (not adjacent in G) [3], [6]. After obtaining the DFS spanning tree, we apply a modification of the ILST algorithm by Salamon [6] to obtain an independence tree all of whose leaves, including the root if it is a leaf, are G-independent. Algorithm 1) is our modification Salamon's ILST:

The MODIFIED-ILST algorithm differs from Salamon's version only on the first line, where we are incorporating a priority function $\rho : V \to \mathbb{Z}$ that determines which of the unvisited neighbors of a vertex will be visited next while building the DFS tree; the neighbor with the highest priority gets visited first.

The priority function gives a ordering heuristic for the depth-first traversal, and a comparison of some choices for this heuristic is the main subject of our study. We experimented with four choices for the ordering heuristic (with two variations for each: forward and reverse).

Algorithm 1 MODIFIED-ILST(G, ρ)

Require: $G = (V, E)$ is a connected graph
Require: $\rho : V \to \mathbb{Z}$ maps each vertex to a unique integer
$T \leftarrow$ DFS-TRAVERSAL(G, ρ)
$r \leftarrow$ the root of T
if T has at least one branching (vertex with degree 2), $\deg_T(r) = 1$, and T has a leaf l s.t. $(r, l) \in E$ **then**
    Add edge (l, r) to T.
    $b \leftarrow$ the branching of T closest to l
    $c \leftarrow$ the child of b in T that is closest to l
    Delete edge (b, c) from T.
end if
return T

The simplest of these heuristics is what we are calling *decreasing degree* (**DD**): the vertices are sorted in decreasing (or non-increasing, to be more accurate) order, so that the vertex with the highest degree gets the highest priority. Ties are broken using an arbitrary value: the vertices' labels, which we have chosen to be unique ordinal numbers. The other four choices are ordering heuristics popularly used in greedy graph coloring algorithms [7]:

- **SL**: *smallest degree last*, also known as reverse degeneracy ordering [8];
- **SLL**: *smallest log-degree last*, introduced by Hasenplaugh *et al*. [7];
- **SD**: *saturation degree*-based ordering [7], [9].

When feasible (*i.e.,* the graph is small enough), we are also considering the optimal CVC cardinality of a graph in our experiments by running a brute-force algorithm.

IV. DATASET

For our experiments, we are using 31 graphs from the Network Repository [10] that are simple (loop-free) and connected. At least 14 of them have degree less than 30, so that the performance of our methods can be compared against optimal minimum connected vertex cover of those graphs, which require exponential-order time to run.

V. EXPERIMENTS

For each graph, we are running the MODIFIED-ILST algorithm with eight priority functions: the four ordering heuristics, once forward and once backward (in reverse order), and we are recording the cardinality of the resulting vertex covers from each heuristic.

For the smaller graphs (order < 30), we are also obtaining the minimum connected vertex cover's cardinality by running a brute-force algorithm. This enables us to see the performance of the heuristics in terms of approximation ratio, too.

VI. RESULTS

Tables VI and VI give the results of these experiments. In these tables, for each graph, the results of the best-performing heuristics are emphasized with boldface font, and the results of the second best–performing heuristics (if there are more than one different results in the graph), with italic font. We make the following observations about the results in :

- The DD-forward ordering provides the best ordering strategy for all but 6 of the 31 graphs used in the experiments.
- With one exception (`biplane-9`), for a graph for where DD-forward does not give the best approximation, it gives the second best performance.
- With few exceptions in the case of the SD heuristic, the forward version of a heuristic always gives at least as good an approximation as its reverse version.

VII. CONCLUSION AND FUTURE DIRECTIONS

As a potentially counter-intuitive result, our results suggest that the DD algorithm, which runs in linear time with only linear additional memory, outperforms the other more sophisticated and more complex vertex-ordering algorithms for finding an approximate solution to the minimum connected vertex problem using DFS spanning trees.

Further investigations may elucidate on why this is the case. It may be fruitful to attempt to establish a causal relationship between the properties of a graph and the best ordering heuristic for finding or approximating its CVC.

Future research with should also make an attempt to compare the results of the CVC-approximation technique studied in this work to those of other papers, such as Fujuto's 2-approximation algorithm that uses maximal matchings [4].

REFERENCES

[1] M. R. Garey and D. S. Johnson, "The Rectilinear Steiner Tree Problem is NP-Complete," *SIAM Journal on Applied Mathematics*, vol. 32, no. 4, pp. 826–834, 1977. Publisher: Society for Industrial and Applied Mathematics.

[2] H. Moser, "Exact Algorithms for Generalizations of Vertex Cover," Master's thesis, Friedrich Schiller University Jena, Munich, Nov. 2005.

[3] C. Savage, "Depth-first search and the vertex cover problem," *Information Processing Letters*, vol. 14, pp. 233–235, July 1982.

[4] T. Fujito and T. Doi, "A 2-approximation NC algorithm for connected vertex cover and tree cover," *Information Processing Letters*, vol. 90, pp. 59–63, Apr. 2004.

[5] Y. Li, W. Wang, and Z. Yang, "The connected vertex cover problem in k-regular graphs," *J Comb Optim*, vol. 38, pp. 635–645, Aug. 2019.

[6] G. Salamon, *Degree-Based Spanning Tree Optimization*. PhD Thesis, Budapest University of Technology and Economics, 2010.

[7] W. Hasenplaugh, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Ordering heuristics for parallel graph coloring," in *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, SPAA '14, (New York, NY, USA), pp. 166–177, Association for Computing Machinery, June 2014.

[8] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *J. ACM*, vol. 30, pp. 417–427, July 1983.

[9] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, pp. 251–256, Apr. 1979.

[10] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015.

TABLE I
COMPARISON OF CVC APPROXIMATION PERFORMANCE OF EACH ORDERING HEURISTIC AGAINST THE OPTIMAL CVC SIZE, FOR SMALL REAL-WORLD GRAPHS

Graph filename (without extension)	$\|V\|$	|CVC|	|Approximate CVC| (and approximation ratio)							
			DD		SL		SLL		SD	
			F	R	F	R	F	R	F	R
Tina_AskCal	11	7	*10 (1.43)*	**9 (1.29)**	**9 (1.29)**	*10 (1.43)*	**9 (1.29)**	*10 (1.43)*	*10 (1.43)*	**9 (1.29)**
Tina_AskCog	11	7	**8 (1.14)**	*9 (1.29)*	**8 (1.14)**	*9 (1.29)*	**8 (1.14)**	*9 (1.29)*	**8 (1.14)**	10 (1.43)
Tina_DisCal	11	8	**9 (1.12)**	*10 (1.25)*	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	*10 (1.25)*
Stranke94	10	9	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**	**9 (1.00)**
Tina_DisCog	11	8	**9 (1.12)**	*10 (1.25)*	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	**9 (1.12)**	*10 (1.25)*
ucidata-gama	16	12	**13 (1.08)**	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	**13 (1.08)**	*14 (1.17)*
soc-tribes	16	12	**13 (1.08)**	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	*14 (1.17)*	**13 (1.08)**	*14 (1.17)*
mammalia-raccoon-proximity	24	20	**21 (1.05)**	23 (1.15)	**21 (1.05)**	*22 (1.10)*	**21 (1.05)**	*22 (1.10)*	**21 (1.05)**	23 (1.15)
bn-mouse_visual-cortex_1	29	13	**15 (1.15)**	17 (1.31)	17 (1.31)	*16 (1.23)*	17 (1.31)	*16 (1.23)*	**15 (1.15)**	17 (1.31)
bio-MUTAG_g1	23	17	**18 (1.06)**	20 (1.18)	*19 (1.12)*	*19 (1.12)*	*19 (1.12)*	*19 (1.12)*	**18 (1.06)**	20 (1.18)
mammalia-primate-association	25	23	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**	**24 (1.04)**
johnson8-2-4	28	21	**23 (1.10)**	*25 (1.19)*	26 (1.24)	26 (1.24)	26 (1.24)	26 (1.24)	26 (1.24)	27 (1.29)
GD02_a	23	10	**12 (1.20)**	16 (1.60)	*13 (1.30)*	16 (1.60)	*13 (1.30)*	16 (1.60)	*13 (1.30)*	14 (1.40)
moreno_sampson	18	15	**16 (1.07)**	*17 (1.13)*	**16 (1.07)**	**16 (1.07)**	**16 (1.07)**	**16 (1.07)**	**16 (1.07)**	*17 (1.13)*

TABLE II
COMPARISON OF CVC APPROXIMATION PERFORMANCE OF ORDERING HEURISTICS FOR LARGE REAL-WORLD GRAPHS

Graph filename (without extension)	$\|V\|$	|Approximate CVC|							
		DD		SL		SLL		SD	
		F	R	F	R	F	R	F	R
gen400-p0-9-55	400	*399*	*399*	*399*	*399*	*399*	*399*	**397**	*399*
SFHH-conf-sensor	403	*349*	400	**347**	401	**347**	401	352	401
G42	2000	**1549**	1982	*1578*	1968	*1578*	1968	1580	1958
G25	2000	**1830**	1958	1887	1992	1887	1992	*1850*	1990
p-hat1500-1	1500	**1452**	1498	**1452**	1498	**1452**	1498	*1459*	1498
DD68	775	**658**	729	699	734	699	734	*684*	741
p-hat1000-2	1000	**968**	999	*971*	999	*971*	999	972	998
G7	800	*779*	786	*779*	798	*779*	798	**775**	798
G9	800	**772**	789	*782*	798	*782*	798	**772**	798
gen200-p0-9-44	200	*199*	*199*	**198**	*199*	**198**	*199*	*199*	*199*
biplane-9	21701	20874	21269	21458	21499	21458	21499	**18370**	*18950*
598a	110971	**101708**	107718	108896	110965	108896	110965	*105702*	110965
ia-hospital-ward-proximity-attr	75	**58**	74	61	74	61	74	*59*	74
polbooks	105	**75**	100	82	101	82	101	*79*	100
G59	5000	**3884**	4949	3966	4915	3966	4915	*3960*	4878
G64	7000	**5421**	6902	*5504*	6868	*5504*	6868	5538	6823
hospital-detailed-list-of-contacts	9518	**127**	*128*	**127**	*128*	**127**	*128*	**127**	*128*