The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

NuMWVC: A Novel Local Search for Minimum Weighted Vertex Cover Problem

Ruizhi Li,[1] Shaowei Cai,[2] Shuli Hu,[3] Minghao Yin,[3*] Jian Gao[4]
[1]School of Management Science and Information Engineering, Jilin University of Finance and Economics, Changchun, 130117, China
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100080, China
[3]School of Computer Science and Information Technology, Northeast Normal University, Changchun, 130117, China
[4] College of Information Science and Technology, Dalian Maritime University, Dalian, 116026, China
lirz111@nenu.edu.cn; caisw@ios.ac.cn; husl903@nenu.edu.cn; ymh@nenu.edu.cn; gaojian@dlmu.edu.cn

Introduction

Given a weighted graph $G = (V, E)$, each vertex v is associated with a weight $w(v)$, the minimum weighted vertex cover (MWVC) problem is to choose a subset of vertices with minimum total weight such that every edge in the graph has at least one of its endpoints chosen. The neighborhood of a vertex v is denoted as $N(v) = \{u \in V | \{u, v\} \in E\}$. The weight of a vertex cover C, is defined as $w(C) = \sum_{v \in C} w(v)$. Each edge e is associated with an edge weight $w_e(e)$. We define the *cost* of a candidate solution C, denoted by $cost(C) = \sum_{cover(e,C)=false} w_e(e)$, which is the total weight of edges uncovered by C. For a vertex v, its score is denoted by $score(v) = (cost(C) - cost(C'))/w(v)$, where $C' = C \backslash \{v\}$ if $v \in C$, and $C' = C \cup \{v\}$ otherwise, which measures the benefit of changing the state of vertex v.

Numerous research efforts have been made on the heuristics for MWVC, such as MS-ITS, proposed in (Zhou et al. 2015), and DLSWCC (Li et al. 2016), clearly dominating other local search algorithms and making a significant improvement. Recent advances in internet have resulted in a collection of massive graphs. However, we are not aware of any work on MWVC for massive graphs. We develop a local search algorithm NuMWVC for MWVC on massive graphs.

Constructing a Vertex Cover with Reductions

The following reduction rules will be used in the constructing procedure to handle vertices with small degrees.

Rule1: If G contains a vertex u s.t. $N(u) = \{v\}$ and $w(u) \geq w(v)$, then there is a MWVC of G that contains v.

Rule2: If G contains a vertex v s.t. $N(v) = \{n_1, n_2\}$, $\{n_1, n_2\} \in E$ and $w(v) \geq w(n_1) + w(n_2)$, then there is an MWVC of G that contains both n_1 and n_2.

Rule3: If G contains a vertex v s.t. $N(v) = \{n_1, n_2\}$, $N(n_1) = \{v, n_2\}$, and $w(v) \geq w(n_1)$, then there is an MWVC of G that contains n_1.

Rule4: If G contains 2 vertices u and v s.t. $N(u) = N(v) = \{n_1, n_2\}$, $\{n_1, n_2\} \notin E$ and $w(u)+w(v) \geq w(n_1)+w(n_2)$, then there is an MWVC of G that contains both n_1 and n_2.

Our local search algorithm starts from a procedure of constructing an initial solution, which is similar to (Cai 2015). First, the proposed reduction rules are applied to put vertices which must be in the optimal solution into C until there is no rule satisfied. We call such kind of vertices *inferred weighted* vertices. After that, the procedure extends C to be a vertex cover of G. Last, redundant vertices are removed from C.

Configuration Checking with Aspiration

The CC strategy (Cai, Su, and Sattar 2011) only allows a vertex v to be added into the current candidate solution if its configuration is changed, but our configuration checking with aspiration (CCA) strategy allows to add v regardless of its configuration if adding it can improve the current best solution. The configuration of a vertex v is a vector consisting of the states of all the vertices in $N(v)$. The vertex selection rules are based on the CCA strategy.

Rmv-Rule: Remove one vertex v, which has the highest $score(v)$ value, and is not an *inferred weighted* vertex.

Add-Rule: If there exist vertices satisfying the aspiration criterion, add one such vertex v with highest $score(v)$ value; otherwise, add one vertex v with highest $score(v)$ value, whose configuration equals to 1.

Self-adaptive Removing (SAR)

Another strategy is a self-adaptive strategy for removing vertices (see Lines 4-7 of Algorithm 1). The local search removes *rmv-num* vertices in each iteration. If the algorithm cannot find a better solution for β iterations, the value of *rmv-num* would be decreased by 1. Since at least one vertex is removed in each step, we guarantee *rmv-num* ≥ 1. In our experiments, α is initialized to 3, and β is set to 50.

The NuMWVC Algorithm

As shown in Algorithm 1, it consists of two stages: the construction stage (Line 1) and the local search stage (Lines 3-14). In each step, the algorithm first removes some vertices from the candidate solution. The number of removed vertices is determined by SAR strategy, while the removed vertices are selected according to the Rmv-Rule. Then NuMWVC iteratively adds a vertex into C according to the Add-Rule until the candidate solution covers all edges, and then the redundant vertices are removed from C. If a better solution is obtained, then C^* is updated by C and the value of *No-improve* is reset to 0; otherwise, *No-improve*++.

*Corresponding author

Algorithm 1: NuMWVC

Input: A graph $G = (V, E, w)$
Output: A vertex cover C of G

Constructing a vertex cover C with reductions;
$C^* \leftarrow C$; *No-improve*$\leftarrow 0$; *rmv-num*$\leftarrow \alpha$;
while *stop criterion is not satisfied* **do**
  if *No-improve* $== \beta$ *and rmv-num* $\neq 1$ **then**
    rmv-num$--$;
  for $i = 0; i <$*rmv-num*$; i++$ **do**
    remove v according to **Rmv-Rule**;
  while C *uncovers some edges* **do**
    add v according to **Add-Rule**;
  remove redundant vertices out of C;
  if $w(C) < w(C^*)$ **then**
    $C^* \leftarrow C$;
    No-improve$\leftarrow 0$;
  else *No-improve*$++$;
  ;
return C^*;

Experimental Evaluation

We carry out experiments to test NuMWVC on real-world massive graphs (Cai 2015). We compare with 2 state-of-the-art algorithms MS-ITS and DLSWCC. For each instance, our algorithm is performed 20 independent runs, and each run is terminated if exceeds 1000s. For sake of space, we only report the average solution ('Avg'), and average time ('Time'). For the minimum solution, NuMWVC dominates the others on almost all instances. If an algorithm fails to give an initial valid solution, the column is marked as "N/A".

For sake of space, we omit the results of some instances, for which DLSWCC and MS-ITS both fail to give an initial valid solution. As results in Table 1, MS-ITS only obtains 27 solutions and NuMWVC obtains better solutions than or same solutions as MS-ITS on these 27 instances. Additionally, NuMWVC obtains better solutions than or same solutions as DLSWCC on 49 instances and DLSWCC only obtains 7 better solutions. For the running time, NuMWVC is obviously faster than others.

Table 1: Experiment results on the massive graphs.

Instance	MS-ITS Avg(Time)	DLSWCC Avg(Time)	NuMWVC Avg(Time)
bio-dmela	149556.8(2126.4)	148540.4(135.4)	**148502.4**(37.7)
bio-yeast	24290.0(53.9)	**24265.0**(3.8)	**24265.0**(1.6)
ca-AstroPh	662926.5(16156.9)	647019.1(420.9)	**645070.6**(543.7)
ca-citeseer	N/A(N/A)	7048225.5(792.4)	**7031461.2**(236.9)
ca-CondMat	704798.5(9860.3)	686344.3(489.2)	**683745.7**(723.7)
ca-CSphd	29609.8(1.2)	**29390.0**(3.7)	**29390.0**(0.4)
ca-dblp-2010	N/A(N/A)	6619251.8(496.9)	**6601770.0**(240.2)
ca-dblp-2012	N/A(N/A)	8986982.4(1106.3)	**8963590.6**(393.3)
ca-Erdos992	28303.0(8.4)	**28298.0**(0.2)	**28298.0**(0.3)
ca-GrQc	122331.5(674.0)	122332.5(95.5)	**122254.3**(35.0)
ca-HepPh	373069.8(10475.3)	365530.8(308.7)	**364001.4**(272.1)
ca-MathSciNet	N/A(N/A)	7668818.0(838.2)	**7637594.9**(306.7)
ia-email-EU	N/A(N/A)	**48269.0**(6.0)	**48269.0**(0.5)
ia-email-univ	32933.0(91.7)	**32931.0**(1.5)	**32931.0**(1.3)
ia-enron-large	N/A(N/A)	695294.8(774.0)	**691651.7**(971.0)
ia-fb-messages	32316.5(44.5)	32300.1(2.3)	**32300.0**(0.7)
ia-reality	**4894.0**(10.4)	**4894.0**(0.0)	**4894.0**(0.0)
ia-wiki-Talk	N/A(N/A)	962194.9(1194.7)	**953135.6**(984.1)
inf-power	121503.3(993.3)	120146.5(110.4)	**120093.7**(37.8)
rec-amazon	N/A(N/A)	2630671.0(1195.3)	**2615387.0**(37.2)
sc-nasasrb	N/A(N/A)	3005889.1(1021.6)	**3000299.7**(999.2)
sc-shipsec1	N/A(N/A)	6844747.6(2056.6)	**6808142.1**(336.9)
soc-brightkite	N/A(N/A)	1187962.3(1162.3)	**1174997.4**(982.4)
soc-delicious	N/A(N/A)	4958206.4(1720.6)	**4926734.0**(996.0)
soc-douban	N/A(N/A)	515288.1(1111.1)	**515270.0**(13.6)
soc-epinions	N/A(N/A)	539915.5(593.2)	**535334.1**(725.1)
soc-gowalla	N/A(N/A)	4729405.5(909.9)	**4713046.2**(169.0)
soc-slashdot	N/A(N/A)	1248151.4(1182.4)	**1233761.6**(997.0)
soc-twitter-follows	N/A(N/A)	**135811.0**(314.7)	**135811.0**(2.5)
socfb-Berkeley13	N/A(N/A)	**1011902.5**(907.1)	1012052.0(504.8)
socfb-CMU	297032.5(1011.9)	**292428.8**(111.0)	292475.6(149.8)
socfb-Duke14	460907.8(1629.4)	450898.3(312.8)	**450884.8**(211.7)
socfb-Indiana	N/A(N/A)	1377961.4(1193.1)	**1373628.2**(690.8)
socfb-MIT	274443.0(12093.2)	**272472.4**(171.2)	272497.1(167.4)
socfb-OR	N/A(N/A)	2116501.0(1169.5)	**2106559.1**(996.8)
socfb-Penn94	N/A(N/A)	1829265.1(1052.8)	**1819186.5**(859.8)
socfb-Stanford3	507561.5(4388.7)	495411.3(479.1)	**495278.8**(294.2)
socfb-UCLA	915068.0(154.9)	**888857.8**(774.8)	889176.7(244.5)
socfb-UConn	793196.8(15843.7)	**771744.5**(638.5)	771968.3(317.1)
socfb-UCSB37	678029.5(5456.8)	**659615.9**(447.5)	659816.9(331.1)
socfb-UIllinois	N/A(N/A)	1417140.9(1197.3)	**1413151.5**(618.8)
socfb-Wisconsin87	N/A(N/A)	**1072009.6**(1081.1)	1072243.4(588.1)
tech-as-caida2007	N/A(N/A)	200755.8(357.5)	**200213.0**(28.6)
tech-internet-as	N/A(N/A)	312308.4(490.5)	**310196.0**(209.2)
tech-p2p-gnutella	N/A(N/A)	918207.3(1058.6)	**916187.8**(906.3)
tech-RL-caida	N/A(N/A)	4204531.8(414.3)	**4199767.5**(993.8)
tech-routers-rf	44936.5(61.0)	44902.3(34.9)	**44894.3**(5.9)
tech-WHOIS	128588.0(6499.9)	128345.3(123.0)	**128336.9**(30.5)
web-arabic-2005	N/A(N/A)	6573003.0(1855.9)	**6557500.9**(584.2)
web-BerkStan	293081.0(2304.9)	286871.4(290.4)	**285547.0**(166.7)
web-edu	79545.5(242.3)	79100.8(47.2)	**79050.3**(48.1)
web-google	**27842.0**(0.8)	**27842.0**(2.6)	**27842.0**(0.8)
web-indochina-2004	409765.0(3187.9)	405773.4(255.3)	**404533.5**(319.9)
web-sk-2005	N/A(N/A)	3135843.5(115.3)	**3134504.5**(134.6)
web-spam	129534.8(644.2)	128994.8(92.6)	**128965.8**(52.7)
web-webbase-2001	144718.5(399.9)	144444.9(186.7)	**144122.0**(44.7)

Acknowledgements

This work is supported by NSFC Grants Nos. 61370156, 61402070, 61503074, 61403076 and 61502464.

References

Cai, S.; Su, K.; and Sattar, A. 2011. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artificial Intelligence* 175(9):1672–1696.

Cai, S. 2015. Balance between complexity and quality: local search for minimum vertex cover in massive graphs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 25–31.

Li, R.; Hu, S.; Zhang, H.; and Yin, M. 2016. An efficient local search framework for the minimum weighted vertex cover problem. *Information Sciences* 372:428–445.

Zhou, T.; Lü, Z.; Wang, Y.; Ding, J.; and Peng, B. 2015. Multi-start iterated tabu search for the minimum weight vertex cover problem. *Journal of Combinatorial Optimization* 1–17.