Information Processing Letters 115 (2015) 667–670

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Approximation algorithms for clique transversals on some graph classes

Min Chih Lin, Saveliy Vasiliev *

CONICET and Instituto de Cálculo, FCEyN, Universidad de Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 19 June 2014
Received in revised form 14 February 2015
Accepted 1 April 2015
Available online 3 April 2015
Communicated by Tsan-sheng Hsu

Keywords:
Approximation algorithms
Clique transversal
Graph classes
NP-hard

ABSTRACT

Given a graph $G = (V, E)$ a *clique* is a maximal subset of pairwise adjacent vertices of V of size at least 2. A *clique transversal* is a subset of vertices that intersects the vertex set of each clique of G. Finding a minimum-cardinality clique transversal is NP-hard for the following classes: planar, line and bounded degree graphs. For line graphs we present a 3-approximation for the unweighted case and a 4-approximation for the weighted case with nonnegative weights; a $\lceil(\Delta(G)+1)/2\rceil$-approximation for bounded degree graphs and a 3-approximation for planar graphs.

1. Introduction

Given a graph $G = (V, E)$ a *clique* is a maximal subset of pairwise adjacent vertices in G of size at least 2. For notational convenience we will also refer as clique to the induced subgraph by such set of vertices. A *clique transversal* is a set of vertices $S \subseteq V$ such that each clique of G intersects S. The Clique-Transversal-Set problem is to find a minimum (cardinality) clique transversal of G. The size of the minimum clique transversal is denoted by $\tau_C(G)$. A closely related parameter is the clique-independence number $\alpha_C(G)$, which is the maximum number of disjoint cliques in G. These parameters are related by the following min-max duality relation: $\alpha_C(G) \leq \tau_C(G)$. If the equality holds for every induced subgraph of G, then G is *clique-perfect*.

The Clique-Transversal-Set problem was studied intensively since the publication [1] by Tuza in 1990. An extremal study of this problem was done in [2] by Erdős, Gallai and Tuza in 1991. The authors proved this problem to be NP-complete for triangle-free graphs and gave a linear time algorithm to find a clique transversal of size at most $n - \sqrt{2n} + \sqrt{2}$ for general graphs. To our best knowledge, it is unknown if this problem is NP for general graphs because the number of cliques in a general graph can be exponential, and we do not know how to efficiently verify if a set of vertices is indeed a clique transversal, and thus for general graphs this problem is considered NP-hard. In 1993 Chang, Farber and Tuza proved this problem to be NP-complete for split graphs and gave a polynomial algorithm for strongly chordal graphs [3]. In the same year, Lee and Chang proved that distance-hereditary graphs are clique-perfect and as a by product obtained a linear time algorithm for computing $\tau_C(G)$ for the same class [4]. In 1996 a polynomial algorithm for comparability graphs was given by Rangan et al. [5]. In 2000, Guruswami and Rangan proved that the problem remains NP-hard when restricted to co-comparability, planar and line graphs [6]. They were the first to attempt the weighted version of clique transversal – they obtained polynomial time algorithms for strongly chordal graphs, chordal graphs of bounded clique size and cographs. They also were the first to solve this problem for a non-clique-perfect class:

* Corresponding author.
E-mail addresses: oscarlin@dc.uba.ar (M.C. Lin), svassiliev@dc.uba.ar (S. Vasiliev).

http://dx.doi.org/10.1016/j.ipl.2015.04.003
0020-0190/

they gave an $O(n^2)$ time algorithm for Helly circular-arc graphs. In 2002, Durán, Lin and Szwarcfiter proved that the problem of deciding whether a set of vertices is a clique transversal is co-NP-complete [7]. In 2008, Durán, Lin, Mera and Szwarcfiter gave a general algorithm for computing a minimum-cardinality clique transversal which runs in $O(n^{2\tau_C(G)})$ time [8]. This algorithm is used to compute a minimum-cardinality clique transversal in $O(n^4)$ time for $\overline{3K_2}$-free circular-arc graphs. They also developed two algorithms for Helly circular-arc graphs: for the weighted case their algorithm runs in $O(n^2)$ and for the unweighted case in linear time. In 2011, Liang and Shan proved that computing $\tau_C(G)$ and $\alpha_C(G)$ is NP-complete for cubic planar graphs of girth 3 [9]. For cubic graphs they gave a 1.96-approximation for computing $\tau_C(G)$.

It is worth noting that with the reduction given by Bertossi [10] the Dominating-Set problem on split graphs is at least as difficult to approximate as the Set-Cover problem. On the other hand, it is easy to see that the size of a minimum-cardinality dominating set equals $\tau_C(G)$ for each split graph G. Thus, the Clique-Transversal-Set problem is at least as difficult to approximate as the Set-Cover problem.

In this work we present constant ratio approximation algorithms for the Clique-Transversal-Set problem for line, planar and bounded degree graphs.

2. Line graphs

In this section we give a 3-approximation algorithm for the Clique-Transversal-Set problem on unweighted line graphs and a 4-approximation algorithm for weighted line graphs with a weight function $w : V \longrightarrow \mathbb{Q}_{\geq 0}$.

In the following, we denote $L(G)$ as the input of the algorithms. Recall that due the algorithm of Roussopoulos [11] we can find $G = (V, E)$ given $L(G)$ in linear time. We assume that G has no isolated edges. Both algorithms described in this section consist in analyzing two types of cliques in $L(G)$: the *vertex-cliques* and *triangle-cliques*, where the former are given by $\delta(v) = \{e \in E : v \text{ is an endpoint of } e\}$ for some $v \in V$, and the latter are triangles in G that will correspond to cliques of size 3 in $L(G)$. These are all the possible cliques in $L(G)$. Indeed, if K_k is clique of $L(G)$ with $k \geq 4$, then it must be associated to a vertex of degree k. If we have a triangle in $L(G)$ which is a clique, then it may be associated to a vertex of degree 3 in G or to a triangle in G. Finally, if we have a K_2 that it is a clique, it must correspond to a vertex of degree 2 in G whose neighbors are non-adjacent.

Let $S \subseteq V$ be the set of vertices that have degree at least 3 or have degree 2 but its neighbors are non-adjacent. For each $v \in S$ it is easy to see that $\delta(v)$ is a clique in $L(G)$. Furthermore, for any $z \in V \setminus S$ we have that $\delta(z)$ is not a clique. To see this, suppose $\deg(z) = 1$ and let u be the neighbor of z. Since we assumed G without isolated edges, u must have another neighbor; therefore $\delta(z) \subset \delta(u)$. If $\deg(z) = 2$, call x and y the neighbors of z, and then $\delta(z) \subset \{zx, zy, xy\}$, and the latter induces a complete subgraph of $L(G)$.

2.1. Unweighted case

Here we strongly exploit the fact that the graph is unweighted and the triangles have three edges. The algorithm consists of two phases: we first transverse all the triangle-cliques and afterwards extend this to transverse all the remaining vertex-cliques.

Start by computing a maximal edge-disjoint set of triangles in G, and take all the edges from these triangles as part of the solution, and call this set of edges F. Observe that we have $k = |F|/3$ edge-disjoint triangles. It is clear that we need at least one edge for each of the selected triangles, since they are edge-disjoint. Let $S' \subseteq S$ be the set of vertices associated to vertex-cliques that were yet not transversed by F. We now introduce a technical result before explaining the second phase of the algorithm.

Lemma 1. *Let $E^* \subseteq E$ be a minimum clique transversal of $L(G)$ and M a maximum matching in $G[S']$. Then,*

$$|S'| - |M| \leq |E^* \setminus F|.$$

Proof. Let $E' \subseteq E$ of minimum cardinality that spans each vertex in S' (that is, every vertex of S' is an endpoint of some edge in E'). Consider the subgraph H with vertex set S' and the edges of E' that have both endpoints in S'. Let M' be a maximum matching in H. Since E' is minimal, it follows that $|S'| = 2|M'| + p$, where $p = |E' \setminus M'|$. On the other hand, we have $|M'| \leq |M|$ and $|E'| = |M'| + p$. Finally, $E^* \setminus F$ spans each vertex of S', and then $|E'| \leq |E^* \setminus F|$. Therefore,

$$\begin{aligned} |S'| &= |E'| + |M'| \leq |E^* \setminus F| + |M'| \\ &\leq |E^* \setminus F| + |M|. \end{aligned}$$ □

In the second phase compute a maximum matching M in $G[S']$, and extend this edge subset in G in a greedy fashion until the subset spans all the vertices in S'. Name the resulting set E'. By definition we have

$$|E'| = |M| + |S'| - 2|M| = |S'| - |M|$$

and that $E' \cup F$ is a clique transversal for $L(G)$. Let E^* be an optimum clique transversal for $L(G)$. We have that

$$\begin{aligned} |E' \cup F| &= |S'| - |M| + |F| = |S'| - |M| + 3k \\ &\leq |E^* \setminus F| + 3|E^* \cap F| \leq 3(|E^* \setminus F| + |E^* \cap F|) \\ &= 3|E^*|. \end{aligned}$$

At this point, we have shown that this algorithm gives a clique transversal of size at most 3 times $\tau_C(L(G))$. Next, we show the algorithm runs in $O(n^{1.5} + m)$ time.

If n and m are the number of vertices and edges of $L(G)$ respectively, we can find G in $O(n + m)$ [11]. Furthermore, we can find all the triangles of G in $O(|E|^{1.5}) = O(n^{1.5})$ using [12]. Clearly, G has no isolated vertices. Therefore, G has n edges and $O(n)$ vertices; thus, we can find a maximum matching in G in $O(\sqrt{|V|}|E|) = O(n^{1.5})$ using [13]. Hence, the resulting running time is $O(n^{1.5} + m)$.

In order to see the tightness of the analysis consider the following scenario. We build the graph G and make the analysis in G and not in $L(G)$. For any integer k take a path P_k, and for each vertex of the path add two neighbors so that a triangle is formed. The algorithm uses $3k$ edges, while an optimum solution uses only k edges, and this may be constructed by taking one edge of each triangle that it is incident to the vertex of P_k.

2.2. Weighted case

In this case we first transverse all the vertex-cliques of $L(G)$. To do so, we solve a minimum weighted edge cover in a graph resulting of adding a new vertex w to G and adding the edges uw for each $u \in V \setminus S$ with weight 0. It is easy to see that the optimum solution to this weighted edge cover has the same value as the best way of spanning the vertices in S. Furthermore, it is trivial to recover a solution to the original problem from the edge cover. Recall that weighted edge cover with nonnegative weights may be solved in polynomial time using the reduction to maximum matching given in [14] (Section 19.3). The weight of this edge cover is smaller than the weight of the optimum clique-transversal, since we covered only a subset of all the cliques.

The second part of the algorithm consists in spanning all the triangle-cliques of $L(G)$ that were not yet transversed. Denote this family $\mathcal{T}$. We are interested in selecting the smallest subset of edges of G that will span all the triangles from $\mathcal{T}$. This is a Set-Cover problem, where the universe of objects is $\mathcal{T}$ and the subsets are $S_e = \{T \in \mathcal{T} : T \text{ contains } e\}$ for $e \in E$. Since each triangle appears exactly in three sets S_e, we can solve this set cover problem with an approximation factor of 3 using the well known frequency approximation algorithm given by Hochbaum in [15].

Taking the union of both sets of edges we get a feasible solution for clique transversal. The weight of edges in the resulting set is at most the sum of the weight of both parts of the algorithm, and thus the overall factor is 4.

3. Planar graphs

In general, for a fixed integer t one can approximate the Clique-Transversal-Set problem with factor t on graphs where the maximum clique has size at most t. This may be achieved by taking a maximal set of vertex-disjoint cliques and outputting all their vertices. This is a clique transversal because of the maximality of the set. On the other hand, each of these cliques requires at least one vertex in order to be transversed, and in our solution we use at most t vertices per clique. Therefore, we have a t-approximation algorithm. By Kuratowski's theorem [16], any planar graph is K_5-free; thus, the above approach leads to a 4-approximation algorithm. In the following, we improve this idea yielding a 3-approximation algorithm for planar graphs.

Begin by searching a maximal disjoint set of cliques of size 2 and 3, and use all their vertices as part of the solution, and call this vertex set S_1. It is clear that all the cliques not yet transversed are of size 4. We look among these cliques a vertex v that belongs only to one non-transversed clique. To see that is always possible to find such vertex consider a planar representation of the graph, and assume for contradiction that there is no such vertex. Take any non-transversed clique of size 4. Clearly, one of its vertices is drawn inside the triangle induced by the rest of the vertices. Name this vertex u. Since there is no vertex that belongs to only one non-transversed clique, there must be another non-transversed clique of size 4 containing u. This clique has its own vertex in the middle which cannot be shared with the previous clique. We can apply this argument infinite times; thus, G has infinite vertices – a contradiction. Now, take any such vertex among the uncovered cliques and use the remaining 3 vertices of that clique as part of the solution. Iterate the same process with the remaining non-transversed cliques of size 4. Call the set of vertices added in the second phase S_2. $S_1 \cup S_2$ is clearly a clique transversal. Furthermore, if in the first phase we considered k_1 disjoint cliques and in the second k_2, then $k_1 + k_2 \leq \tau_C(G)$ and

$$|S_1 \cup S_2| = |S_1| + |S_2| \leq 3k_1 + 3k_2 \leq 3\tau_C(G).$$

Now we show this analysis to be tight. For any integer k consider the path P_{2k+1}, and for each edge of P_{2k+1} add a vertex adjacent to both of its endpoints. The algorithm may consider k disjoint triangles and use all its vertices as part of the solution, and thus the algorithm uses $3k$ vertices. On the other hand, we may construct a clique transversal of cardinality k by only taking half of the vertices from the initial path.

4. Bounded degree graphs

In this section we introduce a $\lceil(\Delta(G)+1)/2\rceil$-approximation algorithm for the Clique-Transversal-Set problem when restricted to bounded degree graphs. The algorithm consists in selecting a maximal vertex-disjoint set of *small* cliques, and uses all their vertices in the solution. If there are still non-transversed cliques, these must be *big*. In order to transverse these big cliques we take one non-transversed clique and select a small subset of its vertices as part of the solution. As long as there are non-transversed cliques we apply the same procedure until every clique is transversed. The approximation ratio is yielded by the definition of *small* and a technical result.

Start by searching a maximal family $\mathcal{F}_1$ of disjoint cliques of size at most $\lceil(\Delta(G)+1)/2\rceil$. Denote V_1 to the vertices in the cliques of $\mathcal{F}_1$. Take V_1 as part of the solution. If all cliques were transversed, the algorithm ends here. Otherwise, let K be a non-transversed clique. By definition, $|K| > \lceil(\Delta(G)+1)/2\rceil$, and K is disjoint with all previously transversed cliques. Take any subset $S \subset K$ of size $\lceil(\Delta(G)+1)/2\rceil$ and add it to the solution.

Lemma 2. *Any non-transversed clique that intersects K must intersect S.*

Proof. For contradiction assume a non-transversed clique C that intersects K but does not intersect S. Since all the cliques of size at most $\lceil(\Delta(G)+1)/2\rceil$ were transversed,

$|C| > \lceil(\Delta(G)+1)/2\rceil$. Take a vertex $v \in K \cap C$. The vertex v is adjacent to each vertex from S and C. Thus,

$$\deg(v) \geq |S| + |C| - 1 > 2\left\lceil \frac{\Delta(G)+1}{2} \right\rceil - 1 \geq \Delta(G). \qquad \square$$

Once K is transversed apply the same procedure if there are still non-transversed cliques. Call the set of cliques used in the second phase of the algorithm $\mathcal{F}_2$ and the vertices of these cliques V_2. Clearly, all these cliques are disjoint by the above lemma. Furthermore, $V_1 \cup V_2$ constitute a clique transversal set. The following holds:

$$|V_1 \cup V_2| = |V_1| + |V_2| \leq \left\lceil \frac{\Delta(G)+1}{2} \right\rceil (|\mathcal{F}_1| + |\mathcal{F}_2|) \leq \left\lceil \frac{\Delta(G)+1}{2} \right\rceil \tau_C(G).$$

The analysis is tight. To see this, it is enough to fix an integer k and consider a graph with two disjoint cliques, $G = K_{\lceil(k+1)/2\rceil} \cup K_{k+1}$. Because of the larger clique we get $\Delta(G) = k$. On the other hand, the algorithm takes all the vertices from the smaller clique and $\lceil(\Delta(G)+1)/2\rceil$ from the bigger. Thus, the solution has $2\lceil(\Delta(G)+1)/2\rceil$ vertices while $\tau_C(G) = 2$.

A possible linear-time implementation of this algorithm is the following. To build $\mathcal{F}_1$ initialize $\mathcal{F}_1 = \emptyset$, $V_1 = \emptyset$, and take any ordering $v_1, \ldots, v_n$ of V. For $i = 1, \ldots, n$ and $v_i \notin V_1$, consider all the subsets of $N(v_i) \cap \{v_{i+1}, \ldots, v_n\} \cap \overline{V_1}$ of size at most $\lceil(\Delta(G)+1)/2\rceil - 1$. If T is one of these subsets and $T \cup \{v_i\}$ is a clique (clearly, v_i is the vertex of this clique with smallest index), then add this clique to $\mathcal{F}_1$ and its vertices to V_1 and continue with the next iteration. Observe that $|N(v_i) \cap \{v_{i+1}, \ldots, v_n\} \cap \overline{V_1}| \leq \deg(v_i)$ and may be obtained in $O(\deg(v_i))$. Let $r_i = \min\{\deg(v_i), \lceil(\Delta(G)+1)/2\rceil - 1\}$. Thus, this procedure yields a running time of

$$O\left(\sum_{i=1}^{n}\left(\deg(v_i) + \sum_{j=1}^{r_i}\binom{\deg(v_i)}{j}(j^2 + j(\deg(v_i) - j))\right)\right) = O\left(\sum_{i=1}^{n} 2^{\Delta(G)}\Delta(G)^2\right) = O(n).$$

To build $\mathcal{F}_2$ the procedure is similar. We avoid considering the already transversed cliques by searching big cliques in $N(v_i) \cap \{v_{i+1}, \ldots, v_n\} \cap \overline{V_1} \cap \overline{V_2}$, for each $v_i \notin V_1 \cup V_2$. The selection of the subset S in each big clique is arbitrary. The running time of this phase is also $O(n)$.

Note that in [9], the authors gave a 1.96-approximation algorithm for cubic graphs. When relaxed to graphs with $\Delta(G) \leq 3$ our $\lceil(\Delta(G)+1)/2\rceil$-approximation for bounded degree graphs yields a 2-approximation.

Acknowledgements

We thank the unknown reviewers for their helpful comments.

The authors were partially supported by UBACyT Grants 20020130100800BA and 20020120100058 and PICT ANPCyT Grants 2010-1970 and 2013-2205.

References

[1] Z. Tuza, Covering all cliques of a graph, Discrete Math. 86 (13) (1990) 117–126.
[2] P. Erdős, T. Gallai, Z. Tuza, Covering the cliques of a graph with vertices, Discrete Math. 108 (13) (1992) 279–289.
[3] G. Chang, M. Farber, Z. Tuza, Algorithmic aspects of neighborhood numbers, SIAM J. Discrete Math. 6 (1) (1993) 24–29.
[4] C.-M. Lee, M.-S. Chang, Distance-hereditary graphs are clique-perfect, Discrete Appl. Math. 154 (3) (2006) 525–536.
[5] V. Balachandran, P. Nagavamsi, C. Rangan, Clique transversal and clique independence on comparability graphs, Inf. Process. Lett. 58 (4) (1996) 181–184.
[6] V. Guruswami, C.P. Rangan, Algorithmic aspects of clique-transversal and clique-independent sets, Discrete Appl. Math. 100 (3) (2000) 183–202.
[7] G. Durán, M.C. Lin, J.L. Szwarcfiter, On clique-transversals and clique-independent sets, Ann. Oper. Res. 116 (1–4) (2002) 71–77.
[8] G. Durán, M.C. Lin, S. Mera, J.L. Szwarcfiter, Algorithms for finding clique-transversals of graphs, Ann. Oper. Res. 157 (1) (2008) 37–45.
[9] Z. Liang, E. Shan, Approximation algorithms for clique-transversal sets and clique-independent sets in cubic graphs, Inf. Process. Lett. 111 (23–24) (2011) 1104–1107.
[10] A.A. Bertossi, Dominating sets for split and bipartite graphs, Inf. Process. Lett. 19 (1) (1984) 37–40.
[11] N.D. Roussopoulos, A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G, Inf. Process. Lett. 2 (4) (1973) 108–112.
[12] N. Chiba, T. Nishizeki, Arboricity and subgraph listing algorithms, SIAM J. Comput. 14 (1) (1985) 210–223.
[13] S. Micali, V.V. Vazirani, An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, in: FOCS, IEEE Computer Society, 1980, pp. 17–27.
[14] A. Schrijver, Combinatorial Optimization – Polyhedra and Efficiency, Springer, 2003.
[15] D.S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, SIAM J. Comput. 11 (3) (1982) 555–556.
[16] C. Kuratowski, Sur le problme des courbes gauches en topologie, Fundam. Math. 15 (1) (1930) 271–283.