Information Processing Letters 116 (2016) 273–278

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A faster FPT algorithm for 3-path vertex cover

Ján Katrenič [a,b,*]

[a] Ness Technologies, Ness Košice Development Center, Slovakia
[b] Institute of Computer Science, P.J. Šafárik University, Košice, Slovakia

ARTICLE INFO

Article history:
Received 18 July 2015
Received in revised form 2 December 2015
Accepted 8 December 2015
Available online 10 December 2015
Communicated by Ł. Kowalik

Keywords:
Fixed-parameter algorithm
Path vertex cover
Dissociation number
Branch and reduce
Analysis of algorithms
Computational complexity
Graph algorithms

ABSTRACT

The k-path vertex cover of a graph G is a subset S of vertices of G such that every path on k vertices in G contains at least one vertex from S. Denote by $\psi_k(G)$ the minimum cardinality of a k-path vertex cover set in G. The minimum k-path vertex cover problem (k-PVCP) is to find a k-path vertex cover of size $\psi_k(G)$. In this paper we present an FPT algorithm to the 3-PVCP with runtime $O(1.8172^s n^{O(1)})$ on a graph with n vertices. The algorithm constructs a 3-path vertex cover of size at most s in a given graph G, or reports that no such 3-path vertex cover exists in G. This improves previous $O(2^s n^{O(1)})$ upper bound by Tu [5] and $O(1.882^s n^{O(1)})$ upper bound by Wu [13].

1. Introduction and motivation

In this paper we consider only finite non-oriented graphs without loops or multiple edges. For a graph G and a positive integer k a subset of vertices $S \subseteq V(G)$ is called a *k-path vertex cover* if every path on k vertices in G contains at least one vertex from S. Denote by $\psi_k(G)$ the minimum cardinality of a k-path vertex cover in G. The *minimum k-path vertex cover problem* (k-PVCP) is problem of finding a k-path vertex cover of size $\psi_k(G)$. Clearly, the 2-PVCP corresponds to the well-known vertex cover problem.

The 3-PVCP is dual problem to *dissociation number problem* (given a graph G, find a maximum size induced subgraph of G with vertex degree at most 1). The dissociation number problem was firstly studied by Papadimitriou and Yannakakis who proved it to be NP-hard in the class of bipartite or planar graphs [10,11]. For an overview on results of polynomially solvable classes of graphs for 3-PVCP we refer the reader to [4].

Recently, approximation algorithms have been designed for 3-PVCP. Kardoš et al. [2] presented polynomial-time 23/11-approximation algorithm to the 3-PVCP and k-approximation to the weighted k-PVCP. This bound for 3-PVCP was improved by Tu and Zhou [8,9] providing 2-approximation algorithm to the weighted 3-PVCP. Tu and Yang [7] proved that 3-PVCP is NP-hard for cubic planar graphs with girth 3, and a 1.57-approximation greedy algorithm was given for 3-PVCP in cubic graphs. Li and Tu [3] gave 2-approximation to 4-PVCP in cubic graphs. An open problem is the existence of a constant c such that, for each $k \geq 2$, k-PVCP has a polynomial-time c-approximation algorithm [1,2].

Kardoš et al. [2] presented an exact algorithm for 3-PVCP with running time of $\mathcal{O}^*(1.5171^n)$ on a graph with n vertices. Throughout this paper, the notation $\mathcal{O}^*()$ suppresses polynomial factors. Recently, Tu [5] gave a fixed-parameter algorithm for 3-PVCP of runtime $\mathcal{O}^*(2^s)$. The algorithm constructs a 3-PVC set of size at most s in a given graph G, or reports that no such 3-PVC set exists

* Correspondence to: Institute of Computer Science, P.J. Šafárik University, Košice, Slovakia.
E-mail address: jkatrenic@gmail.com.

http://dx.doi.org/10.1016/j.ipl.2015.12.002
0020-0190/

in G. Thus, for small parameter values s, such algorithm solves the problem effectively. The result of [5] uses a technique of iterative compression. This result was improved by Wu [13], who presented an algorithm for 3-PVCP of runtime $\mathcal{O}^*(1.882^s)$. Very recently, Tu and Jin [6] presented an algorithm with runtime $O^*(3^s)$ for the 4-PVCP problem.

In this paper, we provide an algorithm for 3-PVCP of runtime $\mathcal{O}^*(1.8172^s)$. Our results are obtained by using a branch-and-reduce approach. The algorithm consists of a set of combinatorial branching and reduction rules. Each rule transforms the current instance of a problem into a set of smaller instances. The analysis of the algorithm considers the worst-case branch over combinatorial cases and derives an upper bound accordingly.

2. Preliminaries

Throughout this paper we use standard graph theory notation. In particular, let $N(v)$ denote the set of all neighbors of v in a graph G. Let $N[v] = N(v) \cup \{v\}$. For a set of vertices S, let $N[S] = \bigcup_{u \in S} N[u]$ and let $N(S) = N[S] \setminus S$. Let $deg_G(v)$ denote the degree of a vertex v in a graph G. The minimum degree of the vertices in a graph G is denoted by $\delta(G)$. Similarly, we write $\Delta(G)$ as the maximum degree of vertices in G. For a graph G and a set of vertices S, let $G - S$ denote the subgraph of G induced by a set of vertices $V(G) \setminus S$. In other words, $G - S$ is obtained from G by deleting all the vertices in S and their incident edges.

For a graph G and a positive integer k a subset of vertices $S \subseteq V(G)$ is called a *k-path vertex cover* (k-PVC) if every path on k vertices in G contains at least one vertex from S. A set of vertices S is called *s-size k-PVC* of G if S is a k-PVC and $|S| \leq s$. Denote by $\psi_k(G)$ the minimum cardinality of a k-path vertex cover in G. Let $\Psi_k(G, s)$ be a boolean function such that $\Psi_k(G, s) = (\psi_k(G) \leq s)$.

Problem 2.1 *(Minimum k-path vertex cover problem, k-PVCP).* Given a graph G, find a k-path vertex cover S of G, $|S| = \psi_k(G)$.

Problem 2.2 *(Parametrized k-path vertex cover problem, k-PPVCP).* Given a graph G and a positive integer s, find a k-path vertex cover S of G, $|S| \leq s$, or report that no such k-path vertex cover exists.

As observed in [5], there is a simple polynomial-time reduction from the minimum k-path vertex problem to the parametrized k-path vertex cover problem. In this paper we focus on the problem of computing $\Psi_k(G, s)$, i.e. the decision version of the problem. It is easy to see that a runtime upper bound for computing $\Psi_k(G, s)$ implies the same runtime upper bound for the k-PPVCP and k-PVCP up to a polynomial factor.

Observe that if a graph G contains a path on vertices $u_1, u_2, \ldots, u_k$ then

$$\Psi_k(G, s) = \bigvee_{i=1,\ldots,k} \Psi_k(G - \{u_i\}, s - 1). \tag{1}$$

Let $T_{path}(n, k)$ be an upper bound for running time of finding a k-path in a given graph on n vertices (k-path problem). The recurrence formula (1) yields to a simple branch-and-reduce algorithm for computing $\Psi_k(G, s)$ with runtime upper bound $\mathcal{O}^*(k^s \cdot T_{path}(|V(G)|, k))$. By result of Zehavi [14] there is a deterministic algorithm for the k-path problem with running time $\mathcal{O}^*(2.597^k)$ and by result of Williams [12] there is a randomized algorithm for the k-path problem with runtime $\mathcal{O}^*(2^k)$. Therefore, the algorithm based on the recurrence formula (1) yields to a fixed-parameter algorithm to the k-PVCP with runtime $\mathcal{O}^*(k^s)$, for every positive constant $k \geq 2$.

3. A faster FPT algorithm for 3-path vertex cover problem

In this section we provide an algorithm for computing the value of $\Psi_3(G, s)$ with runtime $\mathcal{O}^*(1.8172^s)$. Our algorithm is based on the branch-and-reduce approach, which consists of a set of reduction and branching rules. A solution for the current problem instance (G, s) is computed by recursing on smaller subinstances such that if an s-size 3-PVC exists on G, it is computed for at least one subinstance. Moreover, each subinstance (G', s') of the current problem instance (G, s) in our algorithm is such that $s' \leq s$ and G' is a proper induced subgraph of G. If the algorithm considers only one subinstance in a given case then we say it is a reduction rule, otherwise a branching rule. Our algorithm terminates on a trivial problem instance when the graph of the current problem instance is empty or the value of s is negative. Otherwise, the algorithm uses one of the branching or reducing rules.

On a graph G we define the following set of conditions $c_1, \ldots, c_7$:

(c_1): G contains a vertex of degree 0.
(c_2): G has two vertices $u, v \in V(G)$ such that $N(u) \subseteq N[v]$ and $u \neq v$.
(c_3): $\Delta(G) \geq 4$.
(c_4): G has a cycle of length 3.
(c_5): G has two neighboring vertices of degree 2.
(c_6): G contains a vertex of degree 2.
(c_7): G is a cubic graph.

Our algorithm consists of a set of 7 rules. The i-th rule applies when G satisfies the condition c_i but none of conditions $c_1, \ldots, c_{i-1}$. A graph satisfying none of conditions c_1, c_2, c_3 contains only vertices of degree 2 or 3. Note that if a graph contains a vertex of degree 1 it satisfies the condition c_2. Finally, a graph satisfying none of conditions $c_1, \ldots, c_6$ is cubic graph. Therefore at least one of the 7 rules applies on an arbitrary graph G.

Each problem subinstance created by a branching rule in our algorithm is to compute $\Psi_3(G - R, s - |R|)$, for any $R \subseteq V(G)$. In the following lemma we show that if an original problem instance (G, s) does not have a solution, then subinstance $(G - R, s - |R|)$ does not have a solution.

Lemma 3.1. *Let G be a graph and $R \subseteq V(G)$. If $\Psi_3(G, s)$ is false then $\Psi_3(G - R, s - |R|)$ is false.*