Vertex Covers Revisited: Indirect Certificates and FPT Algorithms

CAI Leizhen[*]

Professor Emeritus
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR, China
E-mail:lcai@cse.cuhk.edu.hk

April 24, 2025

Abstract

The classical NP-complete problem VERTEX COVER requires us to determine whether a graph G contains at most k vertices that cover all edges. Despite its intractability, the problem can be solved in Fixed-Parameter Tractable (FPT) time by various techniques when we regard k as a parameter.

In this paper, we introduce six new and simple FPT algorithms for VERTEX COVER parameterized by k. To achieve this, we explore structural properties of vertex covers and leverage these properties to develop FPT algorithms using iterative compression, colour coding, and a novel approach based on the existence of indirect certificates.

In particular, we show that every graph with a k-vertex cover has an indirect certificate consisting of at most $k/3$ vertices. This significant finding lays the groundwork of three new FPT algorithms based on random partition and random selection of vertices in G.

Keywords and phrases: Vertex cover, indirect certificate, FPT algorithm, graph algorithm, randomized algorithm, iterative compression, and colour coding.

[*]Partially supported by General Research Fund 14207118 of the Research Grants Council of Hong Kong. Revision of arXiv:1807.11339 (2018).

1 Introduction

We start with Figure 1 and the following extremely simple algorithm for a graph G, where $N(M)$ denotes the open neighbourhood of marked vertices:

Randomly mark each vertex with probability 1/2 and output $N(M)$.

What problem does the above algorithm solve? Surprisingly, we will show later that it actually finds, with probability at least 4^{-k}, a vertex cover consisting of at most k vertices, if G admits such a vertex cover.

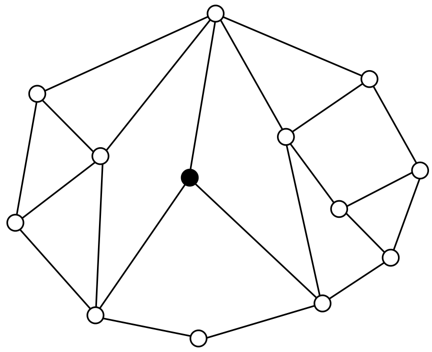

Figure 1: A graph with one dark vertex as certificate.

As for the graph in Figure 1, the single dark vertex in the graph, instead of a vertex cover with eight vertices, can be used as an indirect certificate to certify efficiently that the graph has such a vertex cover. And the existence of such small indirect certificates has an intimate connection with our algorithm in the beginning.

Yes, the subject of this paper is the classical VERTEX COVER problem and we are interested in designing new Fixed-Parameter Tractable (FPT) algorithms for the problem by taking solution size k as parameter.

> VERTEX COVER
> INSTANCE: Graph $G = (V, E)$, and positive integer k as parameter.
> QUESTION: Does G contain a vertex cover of size at most k, i.e., at most k vertices that cover all edges?

Our motivations are multifold though not for the race of fastest FPT algorithms:

- We wish to find structural properties that give us a better understanding of the problem and will be useful for obtaining FPT algorithms.

- Simple FPT algorithms of different flavours for VERTEX COVER are instrumental in teaching FPT algorithms as the problem is almost always used as the first example in such an endeavour.

- It is also intellectually challenging to find different methods to solve a problem, aiming for elegant and simple algorithms and proofs.

There are around a dozen different FPT algorithms for VERTEX COVER [1, 3, 7, 9, 10, 11, 14]. An algorithm of Chen, Kanj, and Xia [7] runs in $O(1.2738^k + kn)$ time, and Harris and Narayanaswamy [11] has recently announced an $O^*(1.25284^k)$-time algorithm. Most such algorithms are based on structural properties of vertex covers. For instance, the basic bounded search tree algorithm [10, 14] is based on the trivial fact that we need to choose at least one end of an edge to cover the edge, and the kernelization algorithm of Buss [3] uses the simple observation that any vertex of degree more than k is forced to be in any vertex cover of size k.

In this paper, we introduce six new and simple FPT algorithms for VERTEX COVER:

- Three FPT algorithms by a novel approach that explores indirect certificates of NP-hard problems and generalizes the random separation method of Cai, Chan and Chan [4] (§2). The foundation of these three algorithms relies on the property that every yes-instance of VERTEX COVER admits an indirect certificate of size at most $k/3$.

- FPT algorithms that rely on characterizations of minimum vertex covers and use the iterative compression method of Reed, Smith and Vetta [19](§3).

- A linear-time algorithm for finding colourful vertex covers in vertex coloured graphs, which yields an FPT algorithm by the colour coding method of Alon, Yuster, and Zwick [2](§4).

In this paper, $G = (V, E)$ is a graph with m edges and n vertices. For convenience, we use *k-vertex cover* for any vertex cover with at most k vertices. Note that $m \leq kn$ for any graph containing a k-vertex cover.

For any subset S of vertices, $N(S)$ denotes the *open neighbourhood* of S, i.e., all vertices in $V - S$ that are adjacent to S, and $N[S]$ the *closed neighbourhood* of S which equals $S \cup N(S)$. By *indirect certificate*, we refer to a structure χ disjoint from a solution X such that χ can be used to certify the existence of a solution in polynomial time, and we say that χ is a *small indirect certificate* if its size is bounded above by a function of parameter k alone. For two disjoint subsets χ and X of vertices V, a partition $(V', V - V')$ of V is a *valid (χ, X)-partition* if $\chi \subseteq V'$ and $X \subseteq V - V'$.

2 Indirect certificates

The random separation method of Cai, Chan and Chan [4] is an innovative method for designing FPT algorithms to solve graph problems. The basic idea of their method is to randomly partition the vertex set of a graph into red and blue vertices to separate a k-*solution* X (i.e., a solution with $\leq k$ vertices) into blue components, and then choose appropriate blue components to form a k-solution. The method is very effective for problems on degree-bounded graphs, and basically can be used to obtain FPT algorithms for any problems dealing with "local properties" [4]. Unfortunately, the original idea of their random separation method requires the size of $N[X]$ be bounded by a function of k, which limits the power of their method to basically degree-bounded graphs. And it seems impossible to obtain an FPT algorithm for VERTEX COVER by random separation.

However, a close examination of the random separation method reveals that it is a special case of a more general approach based on the existence of indirect certificates:

> *First discover a small-size structure χ that is disjoint from a k-solution X and can be used as a certificate for a yes-instance. Then use random partition to produce a valid (χ, X)-partition, and find a k-solution with the aid of χ.*

In connection with this, the random separation method uses $N(X)$ as an indirect certificate χ to cut solution X into pieces (i.e., blue components). In fact two examples in Cai, Chan and Chan [4], including the problem of finding a maximum-weight independent set of size k in planar graphs, implicitly used the above general idea.

This general idea of using indirect certificates seems quite potential for obtaining FPT algorithms, and there have been a few FPT algorithms based on this idea: Cygan et al. [8] have designed an FPT algorithm for obtaining an Eulerian graph by deleting at most k edges, and Cai and Ye [6] have presented FPT algorithms for finding two edge-disjoint (s, t)-paths with some length constraints.

In the following three subsections, we will use this certificate-based approach to design three randomized FPT algorithms, which can be derandomized by standard techniques to yield deterministic FPT algorithms. Foundations of these algorithms are laid by a simple indirect certificate of size at most k (Lemma 2.1) and an elaborated indirect certificate of size at most $k/3$ (Theorem 2.4) for VERTEX COVER.

2.1 Indirect certificates with k vertices

We will show that the algorithm that starts the paper is a randomized FPT algorithm for VERTEX COVER based on indirect certificates. First we present a simple indirect certificate χ with at most k vertices for every yes-instance of VERTEX COVER.

Lemma 2.1 *For any minimal vertex cover X of a graph $G = (V, E)$, G contains an independent set $\chi \subseteq V - X$ with at most $|X|$ vertices such that $N(\chi) = X$.*

Proof. Since X is a minimal vertex cover, every vertex in X covers at least one edge in cut $[X, V - X]$. For each vertex $v \in X$, arbitrarily choose an adjacent vertex $v' \in V - X$ and let $\chi = \{v' : v \in X\}$. Then χ clearly has the property in the lemma as $V - X$ is an independent set. ∎

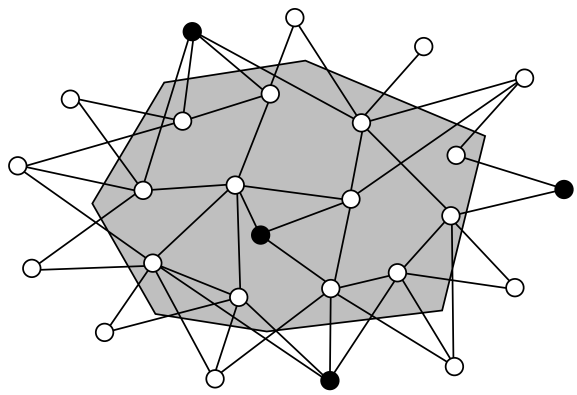

Figure 2: Vertices inside the shaded polygon form a vertex cover X of the graph, and dark vertices χ yield an indirect certificate of size 4. Note that in this instance X is not a minimal vertex cover, and $N(\chi)$ gives a smaller one.

It is straightforward that we can use χ as an indirect certificate to verify that G contains a k-vertex cover in linear time. Surprisingly, this indirect certificate χ also enables us to design a randomized FPT algorithm for VERTEX COVER. In comparison, all NP-complete problems admit direct certificates but such certificates offer no help at all to obtain polynomial algorithms or FPT algorithms.

Let $G = (V, E)$ be a graph that has a k-vertex cover X, and we may assume that X is a minimal vertex cover. By Lemma 2.1, $V - X$ contains at most k vertices χ disjoint from X, and we can obtain X from χ efficiently. As X and χ reside in two disjoint sets, random partition becomes a natural tool to obtain a randomized FPT algorithm for VERTEX COVER.

We randomly and independently colour each vertex by either red or blue with equal probability to form a random partition $\{V_r, V_b\}$ of vertices of G, where V_r and V_b respectively are red and blue vertices. There are two key points to note:

- A random red-blue colouring has probability at least 4^{-k} to produce a valid (χ, X)-partition.
- Once we have a valid (χ, X)-partition, the open neighbourhood of red vertices yields a required vertex cover.

The following is the randomized algorithm at the beginning of the paper rephrased by red-blue colouring, and the algorithm can be derandomized by standard techniques.

> Algorithm VC-IC[k]
> Input: A graph $G = (V, E)$.
> Output: A k-vertex cover X of G, if it exists.
>
> 1. Randomly and independently colour each vertex red or blue with probability 1/2 to generate a random vertex-partition (V_r, V_b) of G.
> 2. Output $N(V_r)$ as X.

Theorem 2.2 *For every yes-instance (G, k) of* VERTEX COVER*, Algorithm VC-IC[k] finds a k-vertex of G with probability at least 4^{-k} and runs in $O(kn)$ time.*

Proof. Let X be a minimal k-vertex cover of G. By Lemma 2.1, G contains at most k vertices χ such that $N(\chi) = X$. Then with probability at least 2^{-k}, Step 1 colours all vertices in χ red and all vertices in X blue. It follows that Step 1 produces a valid (χ, X)-partition (V_r, V_b) with probability at least 4^{-k}.

For any valid (χ, X)-partition (V_r, V_b) of vertices, we see that $N(V_r)$ contains X as all vertices of χ are red and $N(\chi) = X$ by Lemma 2.1. On the other hand, $V_r \subseteq V - X$ and hence $N(V_r)$ contains no vertex of $V - X$ as V_r is an independent set. Therefore $N(V_r) = X$ and the algorithm correctly returns X for such a partition in $O(kn)$ time as $m \leq kn$. ∎

The algorithm can be made into a deterministic FPT algorithm with running time $4^k k^{O(\log k)} n \log n$ by using a family of $(n, 2k)$-universal sets for derandomization [17]. We note that randomized algorithms in the next two subsections can also be derandomized similarly in a standard way.

2.2 Smaller certificates

It is quite surprising that the mere existence of an indirect certificate χ of size k enables us to find in FPT time a k-vertex cover X for yes-instances of VERTEX COVER, even without knowing the actual χ. This motivates us to search for smaller indirect certificates, and indeed we can significantly reduce the size of certificates to $k/3$ only, which leads to an improvement of our FPT algorithm.

We start with a general property regarding a relatively small number of vertices in connection with minimum vertex covers to lay the foundation of certificates of size $k/3$ for VERTEX COVER. For any integer $d \geq 0$, let V_d denote the set of vertices of degree at least d.

Lemma 2.3 *Every graph G with a k-vertex cover contains at most k/d vertices χ, where d is any positive integer, such that for every minimum vertex cover X^* of $G - (N[\chi] \cup V_d(G - N[\chi]))$, vertices $N(\chi) \cup V_d(G - N[\chi]) \cup X^*$ is a minimum vertex cover of G.*

Proof. Let X be a minimum vertex cover of G. We construct χ by choosing at most $k/3$ vertices from $V - X$ as follows. Note that all vertices of X are unmarked initially, and we process vertices $V - X$ in an arbitrary given order. Also note that for any vertex $v \in V - X$, $N(v) \subseteq X$.

> For each $v \in V - X$, we put v into χ and mark all unmarked vertices in $N(v)$ whenever v is adjacent to at least d unmarked vertices in X.

For convenience let $H = G - N[\chi]$ and $G^* = H - V_d(H)$, and note that X^* is a minimum vertex cover of G^*. It is clear that $|\chi| \leq k/d$ as we mark at least d unmarked vertices in X each time we put a vertex into χ. We show that vertices $N(\chi) \cup V_d(H) \cup X^*$ is a minimum vertex cover of G. First we note that $V_d(H) \subseteq X$ as all vertices of $V - X$ in H have degree less than d by the choice of χ. Also, $N(\chi) \cup V_d(H) \cup X^*$ is clearly a vertex cover of G as all edges outside G^* are covered by $N(\chi) \cup V_d(H)$.

It remains to be shown that $N(\chi) \cup V_d(H) \cup X^*$ has the same number of vertices as X. It is helpful to note that X consists of three disjoint parts: $N(\chi)$, $V_d(H)$, and the remaining vertices R. Since $V_d(H) \cup R$ is a minimum vertex cover of H, R is a minimum vertex cover of G^*. Therefore $|R| = |X^*|$ as X^* is also a minimum vertex cover of G^*. It follows that $|N(\chi) \cup V_d(H) \cup X^*| = |X|$ and hence $N(\chi) \cup V_d(H) \cup X^*$ is a minimum vertex cover of G. ∎

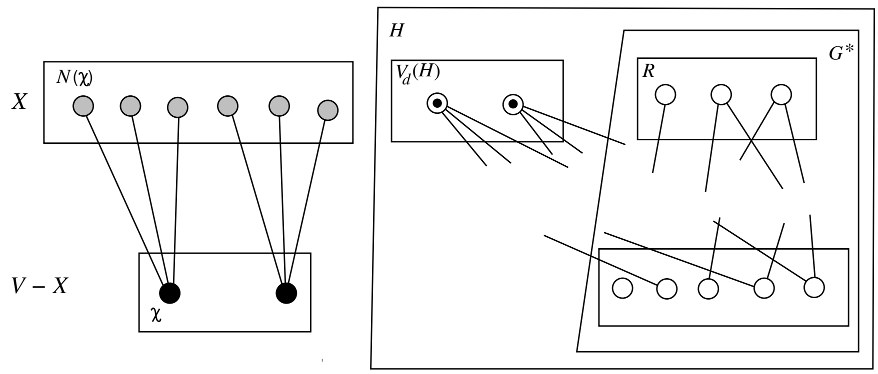

Figure 3: The structure of G with respect to vertex cover X and indirect certificate χ.

The above lemma has interesting and surprising consequences. For VERTEX COVER, Lemma 2.3 for $d = k + 1$ actually gives us an alternative perspective of the kernelization algorithm of Buss [3]: In this case, both χ and $N[\chi]$ are empty, and hence $V_{k+1}(G)$ are vertices forced to be in any k-vertex cover, and graph $G - V_{k+1}(G)$ has maximum degree at most k and forms a kernel of the problem.

Turing to indirect certificates, we recall that VERTEX COVER remains NP-complete for cubic graphs but is trivially solved in linear time for graphs of maximum degree at most two, i.e., graphs consisting of disjoint union of paths and cycles. By setting $d = 3$ in Lemma 2.3, we see that $k/3$ vertices are sufficient to certify yes-instances of VERTEX COVER, which yields the single-vertex certificate in Figure 1 at the beginning of the paper.

Theorem 2.4 *Every yes-instance* (G, k) *of* VERTEX COVER *admits a certificate* χ *with at most* $k/3$ *vertices.*

Proof. Let X be a minimum vertex cover of G. Then $|X| \leq k$ and we use χ in Lemma 2.3 with $d = 3$ as a certificate for our verification algorithm.

Let $G^* = G - (N[\chi] \cup V_3(G - N[\chi]))$. Given G and χ as input, we first compute a minimum vertex cover X^* of G^*, which by Lemma 2.3 yields $N(\chi) \cup V_3(G - N[\chi]) \cup X^*$ as a minimum vertex cover of G. Since G^* is a graph of maximum degree two, the computation of X^* takes linear time. Therefore we can verify that (G, k) is indeed a yes-instance of VERTEX COVER in polynomial time. ∎

With Lemma 2.3 at hand, we can easily improve the success probability of Algorithm VC-IC[k] from 4^{-k} to $2^{-\frac{4}{3}k} > 2.5199^{-k}$. In fact, we can further improve it to 2.1166^{-k} by optimizing the probabilities of colouring vertices red or blue.

Algorithm VC-IC[$k/3$]
Input: A graph $G = (V, E)$.
Output: A k-vertex cover X of G, if it exists.

1. Randomly and independently colour each vertex red with probability 1/4 and blue with probability 3/4 to generate a random vertex-partition (V_r, V_b) of G.
2. Compute a minimum vertex cover X^* of $G - (N[V_r] \cup V_3(G - N[V_r])$, and output $N(V_r) \cup V_3(G - N[V_r]) \cup X^*$ as X.

Theorem 2.5 *For every yes-instance* (G, k) *of* VERTEX COVER, Algorithm VC-IC[$k/3$] *finds a* k*-vertex cover of* G *with probability at least* 2.1166^{-k} *and runs in* $O(kn)$ *time.*

Proof. Let G be a graph with a k-vertex cover X. By Lemma 2.3 for $d = 3$, G contains an indirect certificate χ with at most $k/3$ vertices. Therefore Step 1 produces a valid (χ, X)-partition (V_r, V_b) with probability at least $(1/4)^{k/3}(3/4)^k >$

2.1166^{-k}, implying that the expected number of red-blue colourings to produce a valid (χ, X)-partition is $O(2.1166^k)$.

For a valid (χ, X)-partition, we have $\chi \subseteq V_r$ and $X \subseteq V_b$. Since X cover all edges, we see that V_r is an independent set and hence $N(V_r) \subseteq X$. Now by the choice of χ, no vertex in $V_3(G-N[\chi])$ resides in $V-X$ and hence $V_3(G-N[\chi]) \subseteq X$, which implies $V_3(G - N[V_r]) \subseteq X$ as $V_3(G - N[V_r])$ is an induced subgraph of $V_3(G - N[\chi])$. Finally $G^* = G - (N[V_r] \cup V_3(G - N[V_r]))$ is a graph of maximum degree at most two, i.e., a disjoint union of paths and cycles. Therefore we can easily compute in linear time a minimum vertex cover X^* of G^*. Using the same arguments in the proof of Lemma 2.3, we see that X^*, $N(V_r)$, and $V_3(G - N[V_r])$ together form a minimum vertex cover of G. ∎

2.3 Semi-random partitions

As shown earlier, the existence of indirect certificates for vertex covers enables us to obtain randomized FPT algorithm based on random partition of vertices. Here we further attest the surprising usefulness of indirect certificates by introducing another very simple algorithm for VERTEX COVER based on random selection.

First we note that in Algorithm VC-IC$[k]$, the colour of a vertex is totally independent of colours of other vertices, and the order we process vertices is immaterial. This is in a sense wasteful as a red vertex actually forces all its neighbours to be blue in order to obtain a valid (χ, X)-partition. We now incorporate this forcing step into our algorithm to make the colouring stage semi-random, which increases the chance of obtaining a valid (X, χ)-partition and hence the chance of success for the algorithm.

> Procedure Semi-Random-Partition
> Input: Graph $G = (V, E)$.
> Output: Red-blue partition (V_r, V_b) of V.
>
> Repeat the following until all vertices of G are coloured: Randomly choose an uncoloured vertex v, colour it red or blue with probability p for red and probability $1-p$ for blue, and colour all neighbours of v blue if v is coloured red.

We remark that the above procedure does not recolour neighbours of v as they are either uncoloured or blue when the procedure colours v. To see this, we note that if any vertex $u \in N(v)$ is red, then u is coloured red before v, which would have forced v blue.

Lemma 2.6 *Let p be the probability that a vertex is coloured red. For any vertex v of degree d, the probability $P(v)$ that* Semi-Random-Partition *colours v red and* *all vertices in $N(v)$ blue is at least*

$$\frac{1-(1-p)^{d+1}}{d+1}$$

for $p < 1$ and $1/(d+1)$ for $p = 1$.

Proof. First we note that for a random permutation of $N[v]$, vertex v has probability $1/(d+1)$ to be in position i for any $0 \leq i \leq d$. For v in position i, the i vertices v' of $N(v)$ before v need to be blue in order for v to be red, which forces the remaining vertices in $N(v)$ to receive blue. This gives us success probability of $(1-p)^i p$ for v in position i, as each vertex v' has probability at least $1-p$ to be blue (note that v' may have been forced to be blue before it is processed). Therefore

$$P(v) \geq \frac{1}{d+1}\sum_{i=0}^{d}(1-p)^i p,$$

which yields the closed form in the lemma for $p < 1$ and $1/(d+1)$ for $p = 1$. ∎

We can replace Step 1 in Algorithm VC-IC[k] by Semi-Random-Partition to obtain a new algorithm that finds in linear time a k-vertex cover of G, if it exists, with probability at least $(3/8)^{-k} > 2.667^{-k}$. In fact we can fine-tune the probability p to maximize the success probability of our algorithm, and it turns out that the optimal value is $p = 1$. This is surprising, as it suggests that there is no need for randomness in a red-blue colouring, and it is the ordering of vertices that determines the success probability of our algorithm. Indeed, this perspective gives us another unexpected algorithm based on random selection: *Randomly choose a vertex and declare it to be not in solution.*

Algorithm VC-SRP
Input: A graph $G = (V, E)$.
Output: A k-vertex cover X of G, if it exists.

1. Repeat the following until all vertices are coloured: Randomly and uniformly choose an uncoloured vertex v, colour v red and all neighbours of v blue to form a (V_r, V_b)-partition of V.
2. Output $N(V_r)$ as X.

We now use Lemma 2.1 and Lemma 2.6 to prove the correctness and analyze the success probability of the above algorithm. For an ordering of V, we write $u \prec v$ if vertex u appears before vertex v in the ordering.

Theorem 2.7 *For every yes-instance (G, k) of* VERTEX COVER*, Algorithm* VC-SRP *finds a k-vertex cover of G with probability at least 2^{-k} and runs in $O(kn)$ time.*

Proof. As shown in the proof of Theorem 2.2, $N(V_r)$ is a k-vertex cover once (V_r, V_b) is a valid (χ, X)-partition. To establish the theorem, we show that Semi-Random-Partition has probability at least 2^{-k} to produce a valid (χ, X)-partition (V_r, V_b). First we note that the probability of producing a valid (χ, X)-partition only depends on the subgraph consisting of cut edges $[\chi, X]$. Let G^* be such a subgraph with fewest edges and minimum probability $p(G^*)$ of producing a valid (χ, X)-partition for $|X| = k$, and we examine the structure of G^* to determine $p(G^*)$.

If G^* has two vertices $u, v \in \chi$ sharing a common neighbour w, let $G' = G^* - uw$ and we show that $p(G') \leq p(G^*)$. For this purpose, it suffices to consider a bad vertex ordering of G^* that colours u improperly (i.e., blue) as a consequence of colouring w red. For this to happen, we have $w \prec u$ and w is uncoloured when it is processed. Therefore we also have $w \prec v$ since otherwise w is forced to be blue when v is processed, implying that v is also forced to be blue when w is processed. Therefore the ordering is also a bad vertex ordering for G', which implies that $p(G') \leq p(G^*)$, contradicting the choice of G^*.

Therefore G^* consists of a disjoint union of t stars S_i with vertices χ as centres. By Lemma 2.6, each S_i has probability $1/s_i$ to be properly coloured if S_i contains s_i vertices. Since the colouring of stars S_i is independent, we have

$$p(G^*) = \prod_{1 \leq i \leq t} \frac{1}{s_i}$$

with $\sum_{1 \leq i \leq t}(s_i - 1) \leq k$. It is easy to see that $p(G^*)$ achieves its minimum value 2^{-k} when every $s_i = 2$, i.e., G^* is the disjoint union of k edges. Therefore Semi-Random-Partition has probability at least 2^{-k} to produce a valid (χ, X)-partition (V_r, V_b), and hence the expected number of repeated runs of the algorithm to produce a valid (χ, X)-partition is $O(2^k)$. ∎

We remark that the success probability of Algorithm VC-SRP can be improved to 1.6633^{-k} by using indirect certificates of size $k/3$ and we will not get into details to avoid deviation from the main purpose of the paper.

3 Minimum Vertex Covers

The iterative compression method of Reed, Smith, and Vetta [19] is a useful and powerful tool for designing FPT algorithms. The key idea of the method is to compress a given solution to a smaller one if it can.

We will use the iterative compression method to obtain two FPT algorithms for Vertex Cover. For this purpose, we first turn our attention to determine whether a given vertex cover of $G = (V, E)$ is a minimum vertex cover.

Let X be a set of vertices in G. For a subset $I \subseteq X$, the *outside neighbourhood* of I (w.r.t. X), denoted $N^*(I)$, consists of vertices in $V - X$ that are adjacent to some vertices in I, i.e., $N^*(I) = \bigcup_{v \in I} N(v) - X$. The following characterization of a minimum vertex cover can well be a very old folklore in graph theory, and was observed by the author in 2005 when he was desperately looking for simple examples to teach the iterative compression method.

Lemma 3.1 *A vertex cover X of a graph $G = (V, E)$ is a minimum vertex cover iff for every independent set I of $G[X]$, $|N^*(I)| \geq |I|$.*

Proof. If $G[X]$ contains an independent set I with $|N^*(I)| < |I|$, then $(X - I) \cup N^*(I)$ is a vertex cover of G smaller than X as all edges covered by I are covered by $X - I$ and $N^*(I)$. Conversely, suppose that G has a smaller vertex cover X'. Let $I = X - X'$. Then I is an independent set of $G[X]$, and $N^*(I) \subseteq X' - X$. Since $|X'| < |X|$, we have $|X' - X| < |X - X'|$ and hence $|N^*(I)| < |I|$. ∎

The above lemma enables us to determine in FPT-time whether a $(k+1)$-vertex cover of G is a minimum vertex cover, and obtain a smaller one when it is not. The following compression routine takes $O(2^k kn)$ time as at most 2^{k+1} subsets of X need to be examined in the worst case:

Consider each independent set I in a $(k+1)$-vertex cover X, and check if $|N^*(I)| < |I|$. If so we get a smaller vertex cover $(X - I) \cup N^*(I)$, otherwise G has no k-vertex cover.

As for the question of how to obtain a $(k+1)$-vertex cover in the first place, there are at least three different ways:

1. *Recursively*: Arbitrarily choose a vertex v in G and recursively solve the problem for $G-v$. If $G-v$ has no solution then neither has G. Otherwise, we obtain a k-vertex cover X of $G-v$, which yields a $(k+1)$-vertex cover $X \cup \{v\}$ for G. We need to compress $O(n)$ times, which results in an $O(2^k kn^2)$-time algorithm.

2. *Iteratively*: Order vertices as $v_1, \ldots, v_n$ and let $G_i = G[\{v_1, \ldots, v_i\}]$. Then a k-vertex cover X of G_i yields a $(k+1)$-vertex cover $X \cup \{v_{i+1}\}$ of G_{i+1}, which is then compressed into a k-vertex cover. Again, we get an $O(2^k kn^2)$-time algorithm.

3. *Approximation*: Use a 2-approximation algorithm for vertex covers to find a k'-vertex cover of G first. If $k' > 2k$ then the answer is no, otherwise we compress solutions at most k times, which gives us an $O(4^k k^2 n)$-time algorithm.

Algorithmically, it is not efficient to consider all possible independent sets I inside X, and in fact it is sufficient to consider maximal independent sets only.

Lemma 3.2 *A vertex cover X of a graph $G = (V, E)$ is a minimum vertex cover iff for every maximal independent set I in X, I is a minimum vertex cover of the induced subgraph $G[I \cup N^*(I)]$.*

Proof. If there is a maximal independent set $I \subseteq X$ such that the induced subgraph $G[I \cup N^*(I)]$ admits a vertex cover X' of size smaller than I, then $(X - I) \cup X'$ is a smaller vertex cover of G as $X - I$ covers all edges outside $G[I \cup N^*(I)]$.

Conversely, suppose that G has a vertex cover X' smaller than X. By Lemma 3.1, there is an independent set $I' \subseteq X$ satisfying $|N^*(I')| < |I'|$. Let $I \subseteq X$ be a maximal independent set containing I'. Then $G[I \cup N^*(I)]$ is a bipartite graph as $N^*(I) \subseteq V - X$ is an independent set, and hence admits $(I - I') \cup N^*(I')$ as a vertex cover, which is smaller than I as $|N^*(I')| < |I'|$. ∎

With Lemma 3.2 in hand, we obtain a different compression routine which examines fewer subsets of X:

> Consider each maximal independent set I in a $(k+1)$-vertex cover X, and check if $G[I \cup N^*(I)]$ has a vertex cover X' smaller than I. If so we obtain a smaller vertex cover $(X - I) \cup X'$, otherwise G has no k-vertex cover.

The above compression routine takes $O(3^{k/3} k n^{1.5})$ time as $G[X]$ can contain $O(3^{k/3})$ maximal independent sets in the worst case [16], which can be listed in time $O(3^{k/3}(m + n))$ [20], and it takes $O(m\sqrt{n})$ time to find a minimum vertex cover in a bipartite graph [12]. Recall that $m = O(kn)$ for graphs with k-vertex covers.

4 Colourful vertex covers

Finally we turn to the colour coding method of Alon, Yuster and Zwick [2] for VERTEX COVER. The basic idea of this novel method is to use k colours to colour elements randomly and then try to find a colourful k-solution, i.e., a k-solution whose elements are in distinct colours. If we can find a colourful k-solution in FPT time, then we can find a k-solution in FPT time with probability $k!/k^k > e^{-k}$. The algorithm can be derandomized by a family of perfect hash functions.

The colour coding method is typically useful for parameterized problems whose k-solutions either have a nice structure (e.g., linear, cyclic, tree) or can be partitioned into independent structures of constant size (e.g., triangle packing). Although VERTEX COVER does not seem to possess such properties, we will show that the method is applicable to the problem. In fact, we will present a linear algorithm for the colourful version of VERTEX COVER.

Let $G = (V, E; f)$ be a vertex coloured graph with $f : V \to \{1, \dots, k\}$. We use V_i to denote the set of vertices with colour i, and call each V_i a *colour class.* A vertex cover X of G is *colourful* if all vertices in X have distinct colours, i.e., X contains at most one vertex from each colour class, and we will design a linear algorithm for the following problem:

> COLOURFUL VERTEX COVER
> INSTANCE: Vertex coloured graph G with colours in $\{1, \ldots, k\}$.
> QUESTION: Does G contain a colourful vertex cover?

First we note that the problem is no easier than 2SAT as we can reduce 2SAT to our problem in linear time: For an arbitrary instance (U, C) of 2SAT, we construct a vertex coloured graph G by creating, for each Boolean variable $u_i \in U$, two vertices u_i and $\overline{u}_i$ with colour i and edge $u_i\overline{u}_i$; and adding, for each binary clause $\{x, y\} \in C$, an edge between vertices x and y.

Inspired by the above connection with 2SAT, we first reduce COLOURFUL VERTEX COVER to 2SAT to obtain a quadratic algorithm, and will improve it to a linear algorithm later. For the purpose of a quadratic algorithm, we construct a Boolean formula $\Phi(G)$ for G as follows:

1. For each vertex v, introduce a Boolean variable x_v.
2. For edge set E, let $\Phi(E) = \bigwedge_{uv \in E} x_u \vee x_v$.
3. For each colour class V_i, let $\Phi(V_i) = \bigwedge_{u,v \in V_i \text{ and } u \neq v} \overline{x_u \wedge x_v}$.
4. Set $\Phi(G) = \Phi(E) \bigwedge_{i=1}^{k} \Phi(V_i)$.

Theorem 4.1 *A vertex coloured graph G admits a colourful vertex cover iff its corresponding formula $\Phi(G)$ is satisfiable.*

Proof. Clearly COLOURFUL VERTEX COVER is equivalence to the following integer linear programming: For all $v \in V$, find $x_v \in \{0, 1\}$ to satisfy

$$x_u + x_v \geq 1 \text{ for each edge } uv \text{ of } G\text{, and}$$

$$\sum_{v \in V_i} x_v \leq 1 \text{ for each colour class } V_i \text{ of } G.$$

Note that a vertex v belongs to a colourful vertex cover iff $x_v = 1$.

Now for any $x, y \in \{0, 1\}$, $x + y \geq 1$ is equivalent to $x \vee y$ when we also interpret x and y as Boolean variables. Similarly, $x + y \leq 1$ is equivalent to $\overline{x \wedge y}$. Furthermore, for each colour class V_i, $\sum_{v \in V_i} x_v \leq 1$ is equivalent to $x_u + x_v \leq 1$ for all distinct $u, v \in V_i$. It follows that $\Phi(G)$ is satisfiable iff the above integer linear programming has a solution, and equivalently G admits a colourful vertex cover. ∎

The above theorem enables us to solve COLOURFUL VERTEX COVER in $O(n^2)$ time as $\Phi(G)$ contains $O(n^2)$ binary clauses (note that $\overline{x \wedge y} = \overline{x} \vee \overline{y}$), and 2SAT is solvable in linear time [18]. However, it is unlikely that we can obtain a linear algorithm by reduction to 2SAT as the number of binary clauses seems quadratic for any such reduction. To obtain a linear algorithm, we solve our problem directly by following the idea of limited backtracking for 2SAT.

The key to our linear algorithm is the following forcing properties for edges and colour classes. Recall that for a vertex v, v belongs to a colourful vertex cover iff $x_v = 1$.

1. For an edge, if one end has value 0 then the other end is forced to 1.
2. For a colour class, if one vertex inside it has value 1 then all other vertices in the class are forced to 0.

Therefore the value of a vertex v may force values on other vertices, which may cause a chain reaction to force values on more vertices. Of course, forcing may cause a conflict of values for some vertices. For any $b \in \{0,1\}$ and vertex v, we define $F_b(v)$ to be the set of vertices that are forced to receive a value when $x_v = b$. Assign $F_b(v) = \emptyset$ if $x_v = b$ causes a conflict of value for any forced vertex, which indicates $x_v \neq b$. The following lemma allows us to apply self-reduction to G whenever $F_b(v) \neq \emptyset$ for any $b \in \{0,1\}$.

Lemma 4.2 *For any $b \in \{0,1\}$ and vertex v, G admits a colourful vertex cover iff $G - F_b(v)$ admits one.*

Proof. Let $G' = G - F_b(v)$ and X a colourful vertex cover of G. Since edges of G' can be covered by vertices in G' only, X restricted to G' is clearly a colourful vertex cover of G'.

Conversely, suppose that G' admits a colourful vertex cover X' and let $F' \subseteq F_b(v)$ be vertices with value 1. Then F' is colourful and covers all edges of G outside G', and hence $F' \cup X'$ is a vertex cover of G. For each colour class V_i' in G', $V_i - V_i'$ contains no vertex of F' by the definition of $F_b(v)$. Therefore $F' \cup X'$ is colourful. ∎

The above lemma naturally leads us to the following algorithm which, for clarity, is presented as a parallel algorithm. In our algorithm, we use two processors P_0 and P_1 to compute $F_0(v)$ and $F_1(v)$ independently for a vertex v. For efficiency, we stop processor P_b each time when processor $P_{\overline{b}}$ has obtained a nonempty $F_{\overline{b}}(v)$. Our algorithm can be converted into a sequential one by the standard dovetailing technique.

Algorithm Colourful-VC
Input: A vertex coloured graph $G = (V, E; f)$ with $f : V \to \{1, \ldots, k\}$.
Output: A colourful vertex cover X of G.

while G contains a vertex v with no value **do** the following in parallel:
Processor P_0: compute $F_0(v)$;
    if $F_0(v) \neq \emptyset$ **then stop** P_1,
        assign values to $F_0(v)$ accordingly,
        and $G \leftarrow G - F_0(v)$;
Processor P_1: compute $F_1(v)$;
    if $F_1(v) \neq \emptyset$ **then stop** P_0,
        assign values to $F_1(v)$ accordingly,
        and $G \leftarrow G - F_1(v)$;
if both $F_0(v)$ and $F_1(v)$ are empty **then** return "No solution" and **halt**;
end while;
Return all vertices with value 1 as X.

We can compute $F_b(v)$ by BFS with one of the following two actions in extending the current vertex u to other vertices. The action depends on the value of the current vertex u.

Case $x_u = 0$: **for** each edge uu' **do**
    if $x_{u'} = 0$ **then** $F_b(v) \leftarrow \emptyset$ and **stop**
        else $x_{u'} \leftarrow 1$.
Case $x_u = 1$: **for** every other vertex u' in the colour class containing u **do**
    if $x_{u'} = 1$ **then** $F_b(v) \leftarrow \emptyset$ and **stop**
        else $x_{u'} \leftarrow 0$.

Theorem 4.3 *For any vertex coloured graph G with k colours, it takes $O(kn)$ time to find a colourful vertex cover in G, if it exists.*

Proof. The correctness of algorithm Colourful-VC follows from Lemma 4.2. If both $F_0(v)$ and $F_1(v)$ are empty, then the algorithm clearly halts in time $O(m+n)$ as it just performs BFS twice. Otherwise one of them is not empty and we may assume that, without loss of generality, $F_0(v)$ is not empty and processor P_0 finishes before P_1. Let $T(G)$ denote the time of the algorithm for processing graph G, and we have the following recurrence:

$$T(G) = T(G - F_0(v)) + O(\text{the number of edges covered by } F_0(v))$$

since the latter term is the time spent by processor P_0 to compute $F_0(v)$. It is obvious by induction that $T(G) = O(m + n) = O(kn)$ as $m = O(kn)$ for graphs with k-vertex covers. ∎

The above theorem implies that the colour coding method can be used to solve VERTEX COVER in FPT time. We remark that the problem becomes intractable if we wish to minimize the size of X, or allow X to contain at most two vertices from each colour class.

5 Concluding remarks

We have presented six new and simple FPT algorithms for the classical VERTEX COVER by using iterative compression, colour coding, and a new approach based on indirect certificates. These algorithms explore structural properties of the problem, which deepens our understanding of the problem. In particular, the existence of indirect certificates of size $k/3$ for VERTEX COVER is quite interesting and surprising. We hope that ideas in the paper will be useful in designing FPT algorithms for other problems — although we have to do exhaustive search one way or another, it is fun and challenging to do such bad things in clever ways.

The certificate-based approach seems quite potential for designing FPT algorithm, though so far several problems only are solved by the method. Of course, indirect certificates are interesting in their own right. In particular, certificates of size a fraction of solution size have both theoretical and practical values. We note that certificates of size $k/3$ for VERTEX COVER implies that if there is a polynomial reduction from a problem Π to VERTEX COVER that preserves solution size k, then Π also has certificates of size $k/3$. Using this reduction approach, we can infer that the following two NP-hard problems both have certificates of size at most $k/3$: finding a truth assignment with at most k 1's for 2SAT [15], and deleting k edges to destroy all alternating paths in an edge coloured graph [5].

Problem 5.1 *What kinds of problems have certificates whose size is a fraction of solution size?*

In connection with FPT algorithms, it is not hard to propose various small indirect certificates χ. In fact, it is easy to argue through kernelization that all FPT problems admit small indirect certificates, but it is not clear whether small indirect certificates imply FPT algorithms. A pressing question is how to use an indirect certificate χ properly to find X, and the main obstacle seems to be false positive noises caused by red elements not from χ.

Problem 5.2 *For a valid (χ, X)-partition, under what conditions can we find a k-solution in FPT time?*

Coming back to VERTEX COVER, we can reduce the size of certificates from $k/3$ to $k/3 - c$ for any constant c, but it seems unlike to shave off $\log k$ from $k/3$ as in the $2k - c \log k$ kernelization [13] of the problem. The author feels that $k/3$ is best possible fractionally.

Conjecture 5.3 *For some $d > 3$, if every yes-instance of* VERTEX COVER *has a certificate of size k/d then* P = NP.

Finally, in echo to the algorithm that starts the paper, we conclude with the following simple algorithm that uses random orientation to solve with probability at least 2^{-k} the related PARTIAL VERTEX COVER problem: Find fewest vertices in a graph G to cover at least k edges.

> Randomly orient each edge to obtain a digraph $\vec{G}$ from G. Order vertices $\vec{G}$ nonincreasingly by their out-degrees, and choose vertices until the sum of their out-degrees is at least k.

References

[1] Abu-Khzam, F. N., Collins, R. L., Fellows, M. R., Langston, M. A., Suters, W. H., and Symons, C. T., Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments, ALENEX/ANALC, 69 (2004).

[2] Alon, N., Yuster, R., and Zwick, U., Color-coding, Journal of the ACM 42(4), 844–856 (1995).

[3] Buss, J. F. and Goldsmith, J., Nondeterminism within P. SIAM Journal on Computing 22(3), 560-572 (1993)

[4] Cai, L., Chan, S. M., and Chan, S. O., Random separation: A new method for solving fixed-cardinality optimization problems. In International Workshop on Parameterized and Exact Computation, LNCS 4169, pp. 239-250 (2006).

[5] Cai, L., and Leung, O. Y., Alternating path and coloured clustering, arXiv:1807.10531 (2018).

[6] Cai, L. and Ye, J., Finding two edge-disjoint paths with length constraints. In International Workshop on Graph-Theoretic Concepts in Computer Science, LNCS 9941 pp. 62-73 (2016).

[7] Chen, J., Kanj, I. A., and Xia, G., Improved upper bounds for vertex cover. Theoretical Computer Science, 411(40-42), 3736-3756 (2010).

[8] Cygan, M., Marx, D., Pilipczuk, M., Pilipczuk, M., and Schlotter, I., Parameterized complexity of Eulerian deletion problems. Algorithmica, 68(1), 41-61 (2014).

[9] Dehne, F., Fellows, M., Rosamond, F., and Shaw, P., Greedy localization, iterative compression, and modeled crown reductions: New FPT techniques, an

improved algorithm for set splitting, and a novel 2k kernelization for Vertex Cover. In International Workshop on Parameterized and Exact Computation, LNCS 3162 pp. 271-280 (2004).

[10] Downey, R. G., and Fellows, M. R., Parameterized complexity. Springer Science & Business Media (2012).

[11] Harris, D. G. and Narayanaswamy, N. S., A faster algorithm for vertex cover parameterized by solution size. arXiv preprint arXiv:2205.08022 (2022).

[12] Hopcroft, J. E., and Karp, R. M., An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on computing, 2(4), 225-231 (1973).

[13] Lampis, M., A kernel of order $2k - c \log k$ for vertex cover. Information Processing Letters, 111(23-24), 1089-1091 (2011).

[14] Mehlhorn, K., Data Structures and Algorithms 2: Graph Algorithms and NP-completeness (Vol. 2). Springer Science & Business Media (1984).

[15] Misra, N., Narayanaswamy, N. S., Raman, V., and Shankar, B. S., Solving MINONES-2-SAT as fast as vertex cover. In International Symposium on Mathematical Foundations of Computer Science, LNCS 6281 pp. 549-555 (2010).

[16] Moon, J. W., and Moser, L., On cliques in graphs. Israel journal of Mathematics, 3(1), 23-28 (1965).

[17] Naor, M., Schulman, L. J., and Srinivasan, A., Splitters and near-optimal derandomization. In Proc. 36th Annual Symposium on Foundations of Computer Science, pp. 182-191 (1995).

[18] Petreschi, R., and Simeone, B., Experimental comparison of 2-satisfiability algorithms. RAIRO-Operations Research, 25(3), 241-264 (1991).

[19] Reed, B., Smith, K., and Vetta, A., Finding odd cycle transversals. Operations Research Letters, 32(4), 299-301 (2004).

[20] Tomita, E., Tanaka, A., and Takahashi, H., The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science, 363(1), 28-42 (2006).