Paths, Flowers and Vertex Cover*

Venkatesh Raman† M. S. Ramanujan† Saket Saurabh†

Abstract

It is well known that in a bipartite (and more generally in a König) graph, the size of the minimum vertex cover is equal to the size of the maximum matching. We first address the question whether (and if not when) this property still holds in a König graph if we insist on forcing one of the two vertices of some of the matching edges in the vertex cover solution. We characterize such graphs using the classical notions of augmenting paths and flowers used in Edmonds' matching algorithm. Using this characterization, we develop an $O^*(9^k)$[1] algorithm for the question of whether a general graph has a vertex cover of size at most $m+k$ where m is the size of the maximum matching. Our algorithm for this well studied Above Guarantee Vertex Cover problem uses the technique of iterative compression and the notion of important separators, and improves the runtime of the previous best algorithm that took $O^*(15^k)$ time. As a consequence of this result we get that well known problems like Almost 2-SAT (deleting at most k clauses to get a satisfiable 2-SAT formula) and König Vertex Deletion (deleting at most k vertices to get a König graph) also have an algorithm with $O^*(9^k)$ running time, improving on the previous bound of $O^*(15^k)$.

1 Introduction

The classical notions of *matchings* and *vertex covers* have been at the center of serious study for several decades in the area of Combinatorial Optimization [14]. In 1931, König and Egerváry independently proved a result of fundamental importance: in a bipartite graph the size of a maximum matching equals that of a minimum vertex cover [14]. This led to a polynomial-time algorithm for finding a minimum vertex cover in bipartite graphs. Interestingly, this min-max relationship holds for a larger class of graphs known as König-Egerváry graphs and it includes bipartite graphs as a proper subclass. König-Egerváry graphs will henceforth be called König graphs. Our first result in this paper is an extension of this classical result. That is, we address the following question:

> *When does a König graph have a minimum vertex cover equal to the size of a maximum matching when we insist on forcing one of the two vertices of some of the matching edges in the vertex cover solution?*

We resolve this problem by obtaining a excluded-subgraph characterization for König graphs satisfying this property. More precisely, let G be a König graph, M be a maximum matching

*A preliminary version of this paper appeared in the proceedings of ESA 2011.

†The Institute of Mathematical Sciences, Chennai, India. {vraman|msramanujan|saket}@imsc.res.in

[1] O^* notation suppresses polynomial factors

of G and let X be a set of vertices containing exactly one vertex from some of the edges of M. Then G has a minimum vertex cover of size $|M|$ containing X if and only if it does not contain "certain kind of M-augmenting paths and M-flowers." These notions of augmenting paths and flowers are the same as the one used in the classical maximum matching algorithm of Edmonds [8] on general graphs.

Our algorithmic motivation for this excluded-subgraph characterization stems from obtaining a faster algorithm for a version of the Vertex Cover problem studied in parameterized complexity. For decision problems with input size n, and a parameter k, the goal in parameterized complexity is to design an algorithm with runtime $f(k)n^{O(1)}$ where f is a function of k alone, as contrasted with a trivial $n^{k+O(1)}$ algorithm. Such algorithms are said to be fixed parameter tractable (FPT). We also call an algorithm with a runtime of $f(k)n^{O(1)}$, as an FPT algorithm, and such a runtime as FPT runtime. The theory of parameterized complexity was developed by Downey and Fellows [7]. For recent developments, see the book by Flum and Grohe [9]. The version of classical Vertex Cover that we are interested in is following.

Above Guarantee Vertex Cover (agvc)

Input:	$(G = (V, E), M, k)$, where G is an undirected graph, M is a maximum matching for G, k a positive integer
Parameter:	k
Question:	Does G have a subset S of size at most $\|M\| + k$ that covers all the edges?

Prior to this work, the only known parameterized algorithm for agvc was using a parameter preserving reduction to Almost 2-SAT. In Almost 2-SAT, we are given a 2-SAT formula ϕ, a positive integer k and the objective is to check whether there exists at most k clauses whose deletion from ϕ can make the resulting formula satisfiable. The Almost 2-SAT problem was introduced in [15] and a decade later it was shown by Razgon and Barry O'Sullivan [21] to have an $O^*(15^k)$ time algorithm, thereby proving fixed-parameter tractability of the problem when k is the parameter. The Almost 2-SAT problem is turning out to be a fundamental problem in the context of designing parameterized algorithms. This is evident from the fact that there is a polynomial time parameter preserving reduction from problems like Odd Cycle Transversal [11] and agvc [19] to it. An FPT algorithm for Almost 2-SAT led to FPT algorithms for several problems, including agvc and König Vertex Deletion [19]. In recent times this has been used as a subroutine in obtaining a parameterized approximation as well as an FPT algorithm for Multi-Cut [17, 18]. Our second motivation for studying agvc is that it also implies a faster FPT algorithm for Almost 2-SAT. This is obtained through a parameter preserving reduction from Almost 2-SAT to agvc and hence this also shows that these two problems are equivalent.

The standard version of Vertex Cover, where we are interested in finding a vertex cover of size at most k for the given parameter k was one of the earliest problems that was shown to be FPT [7]. After a long race, the current best algorithm for Vertex Cover runs in time $O(1.2738^k + kn)$ [2]. However, when $k < m$, the size of the maximum matching, the standard version of Vertex Cover is not interesting, as the answer is trivially NO. And if m is large (suppose, for example, the graph has a perfect matching), then for the cases the

problem is interesting, the running time of the standard version is not practical, as k, in this case, is quite large. This also motivates the study of AGVC.

Our results and methodology. Many of the recent FPT algorithms, including the ones for ALMOST 2-SAT [21], DIRECTED FEEDBACK VERTEX SET [4], MULTIWAY CUT [3], MULTICUT [18] are based on a combination of the method of iterative compression introduced in [22] and graph separation. In the iterative compression method, we assume that a solution of size $k+1$ is part of the input, and attempt to compress it to a solution of size k. The method adopted usually is to begin with a subgraph that trivially admits a $(k+1)-$sized solution and then expand it iteratively. The main ingredient of the graph separation part of these algorithms is to find the right structures to eliminate, which in most of these cases are certain kind of paths, and to be able to eliminate them in FPT time. Notions of "important sets" and "important separators" have turned out to be very useful in these cases [3, 16, 18]. We follow the same paradigm here and using our excluded subgraph characterization find a set of structures that we need to eliminate to solve AGVC faster. More precisely, using our graph theoretic results together with the algorithmic technique of iterative compression and the notion of important separators, we develop an $O^*(9^k)$ algorithm for AGVC. This improves the runtime of the previous best algorithm that took $O^*(15^k)$ time. This in turn together with known parameterized reductions implies faster algorithms $(O^*(9^k))$[2] for a number of problems including ALMOST 2-SAT (both variable and clause variants) and KÖNIG VERTEX DELETION.

Organization of the paper. In Section 3 we give a general outline of our algorithm and describe the method of iterative compression applied to the AGVC problem. In Section 4 we show that the structures we need to eliminate in the case of the AGVC problem are precisely the augmenting paths and flowers seen in the classical maximum matching algorithm of Edmonds [8] on general graphs. In Section 5, we then exploit the structure given by the combinatorics of vertex covers and maximum matchings to obtain an FPT algorithm for AGVC that runs in time $O^*(9^k)$. Finally, in Section 5.3 we prove that ALMOST 2 SAT has an $O^*(9^k)$ algorithm by giving a polynomial time parameter preserving reduction from AGVC to ALMOST 2 SAT. We conclude with some remarks and give a brief overview of the subsequent research on these problems in Section 6.

2 Preliminaries

Let $G = (V, E)$ be a graph and $D = (V, A)$ be a directed graph. We call the ordered pair $(u, v) \in A$ arc and the unordered pair $(u, v) \in E$ edge. For a subset S of V, the *subgraph of G induced by S* is denoted by $G[S]$. By $N_G(u)$ we denote (open) neighborhood of u that is set of all vertices adjacent to u. Similarly, for a subset $T \subseteq V$, we define $N_G(T) = (\cup_{v \in T} N_G(v)) \setminus T$. Given a graph $G = (V, E)$ and two disjoint vertex subsets V_1, V_2 of V, we let (V_1, V_2) denote the bipartite graph with vertex set $V_1 \cup V_2$ and edge set $\{(u, v) : (u, v) \in E \text{ and } u \in V_1, v \in V_2\}$. Given a graph G, we use $\mu(G)$ and $\beta(G)$

[2]Subsequent work on improved algorithms for ALMOST 2 SAT and related problems is summarized in the last section.

to denote, respectively, the size of a maximum matching and a minimum vertex cover. A graph $G = (V, E)$ is said to be *König* if $\beta(G) = \mu(G)$. If M is a matching and $(u, v) \in M$ then we say that *u is the partner of v in M*. If the matching being referred to is clear from the context we simply say *u is a partner of v*. The vertices of G that are the endpoints of edges in the matching M are said to be *saturated by M* and we denote the set of these vertices by $V(M)$; all other vertices are *unsaturated by M*. Given graph $G = (V, E)$, and a (not necessarily simple) path $P = v_1, \ldots, v_t$, we define by $Rev(P)$ the path $v_t, v_{t-1}, \ldots, v_1$. Even though it may not make sense to talk about the direction of a path in an undirected graph, we will use this notation to simplify our presentation. We will call the number of edges in P the length of P and represent it by $|P|$. Let $P_1 = v_1, \ldots, v_t$ and $P_2 = v_t, \ldots, v_x$ be two paths which have only the vertex v_t in common. We represent by $P_1 + P_2$ the concatenated path $v_1, \ldots, v_t, v_{t+1}, \ldots, v_x$. We also need the following characterization of König graphs.

Lemma 1 (see, for example, [19])**.** *A graph $G = (V, E)$ is König if and only if there exists a bipartition of V into $V_1 \uplus V_2$, with V_1 a vertex cover of G such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1.*

Definition 1. *Let Z be a finite set. A function $f : 2^Z \to \mathbb{N}$ is submodular if for all subsets A and B of Z, $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$*

3 Outline of the Algorithm

We first make use of a known reduction that allows us to assume that the input graph has a perfect matching. Given an instance $(G = (V, E), M, k)$ of AGVC, in polynomial time we obtain an equivalent instance with a perfect matching using [19, Lemma 5]. That is, if (G, M, k) is an instance of AGVC and G is a graph without a perfect matching, then in polynomial time we can obtain an instance (G', M', k) such that G' has a perfect matching M' and (G, M, k) is a YES instance of AGVC if and only if (G', M, k) is a YES instance of AGVC. Because of this reduction, throughout this paper we assume that in our input instance (G, M, k), the matching M is a perfect matching of G. We now describe the iterative compression step, which is central to our algorithm, in detail.

Iterative Compression for AGVC. Given an instance $(G = (V, E), M, k)$ of AGVC let $M = \{m_1, \ldots, m_{n/2}\}$ be a perfect matching for G where $n = |V|$. Define $M_i = \{m_1, \ldots, m_i\}$, $m_i = (a_i, b_i)$, and $G_i = G[V(M_i)]$, $1 \leq i \leq \frac{n}{2}$. We iterate through the instances (G_i, M_i, k) starting from $i = k + 1$ and for the i^{th} instance, with the help of a *known* solution S_i of size at most $|M_i| + k + 1$ we try to find a solution $\hat{S}_i$ of size at most $|M_i| + k$. Formally, the compression problem we address is as follows.

ABOVE GUARANTEE VERTEX COVER COMPRESSION (AGVCC)

Input:	$(G = (V, E), S, M, k)$, where G is an undirected graph, M is a perfect matching for G, S is a vertex cover of G of size at most $\|M\| + k + 1$, k a positive integer
Parameter:	k
Question:	Does G have a vertex cover $\hat{S}$ of size at most $\|M\| + k$?

We will reduce the AGVC problem to $\frac{n}{2} - k$ instances of the AGVCC problem as follows. Let $I_i = (G_i, M_i, S_i, k)$ be the i^{th} instance. Clearly, the set $V(M_{k+1})$ is a vertex cover of size at most $|M_{k+1}| + k + 1$ for the instance I_{k+1}. It is also easy to see that if $\hat{S}_{i-1}$ is a vertex cover of size at most $|M_{i-1}| + k$ for instance I_{i-1}, then the set $\hat{S}_{i-1} \cup V(m_i)$ is a vertex cover of size at most $|M_i| + k + 1$ for the instance I_i. We use these two observations to start off the iteration with the instance $(G_{k+1}, M_{k+1}, S_{k+1} = V(M_{k+1}), k)$ and look for a vertex cover of size at most $|M_{k+1}| + k$ for this instance. If there is such a vertex cover $\hat{S}_{k+1}$, we set $S_{k+2} = \hat{S}_{k+1} \cup V(m_{k+2})$ and ask for a vertex cover of size at most $|M_{k+2}| + k$ for the instance I_{k+2} and so on. If, during any iteration, the corresponding instance does not have a vertex cover of the required size, it implies that the original instance is also a NO instance. Finally the solution for the original input instance will be $\hat{S}_{\frac{n}{2}}$. Since there can be at most $\frac{n}{2}$ iterations, the total time taken is bounded by $\frac{n}{2}$ times the time required to solve the AGVCC problem.

Our algorithm for AGVCC is as follows. Let the input instance be $I = (G = (V, E), S, M, k)$. Let M' be the edges in M which have both vertices in S. Note that $|M'| \leq k + 1$. Then, $G \setminus V(M')$ is a König graph and by Lemma 1 has a partition (A, B) such that A is a minimum vertex cover and there is a matching saturating A across the cut (A, B), which in this case is $M \setminus M'$. We guess a subset $Y \subseteq M'$ with the intention of picking both vertices of these edges in our solution. For the remaining edges of M', exactly one vertex from each edge will be part of our eventual solution. For each edge of $M' \setminus Y$, we guess the vertex which is not going to be part of our eventual solution. Let the set of vertices guessed this way as not part of the solution be T. Define $L = A \cap N_G(T)$ and $R = B \cap N_G(T)$. Clearly our guess forces $L \cup R$ to be part of the solution. We have thus reduced this problem to checking if the instance $(G[V(M \setminus M')], A, M \setminus M', k - |M'|)$ has a vertex cover of size at most $|M \setminus M'| + k - |M'|$ which contains L and R. We formally define this annotated variant as follows.

ANNOTATED ABOVE GUARANTEE VERTEX COVER (A-AGVC)

Input:	$(G = (A, B, E), M, L, R, k)$, where G is an undirected König graph, (A, B) is a partition of the vertex set of G, A is a minimum vertex cover for G, M is a perfect matching for G saturating A and B, $L \subseteq A$ and $R \subseteq B$, k a positive integer
Parameter:	k
Question:	Does G have a vertex cover of size at most $\|M\| + k$ such that it contains $L \cup R$?

Our main result is the following Lemma.

Lemma 2. A-AGVC *can be solved in* $O^*(4^k)$ *time.*

Given Lemma 2 the running time of our algorithm for AGVCC is bounded as follows. For every $0 \leq i \leq k$, for every i sized subset Y, for every guess of T, we run the algorithm for A-AGVC with parameter $k - i$. For each i, there are $\binom{k+1}{i}$ subsets of M' of size i, and for every choice of Y of size i, there are 2^{k+1-i} choices for T and for every choice of T, running the algorithm for A-AGVC given by Lemma 2 takes time $O^*(4^{k-i})$. Hence, the running time of our algorithm for AGVCC is bounded by $O^*(\Sigma_{i=0}^{k}\binom{k+1}{i}2^{k+1-i}4^{k-i}) = O^*(9^k)$ and hence our algorithm for AGVC runs in time $O^*(9^k)$. Thus we have the following Theorem.

Theorem 1. AGVC *can be solved in* $O^*(9^k)$ *time.*

Sections 4 and 5 are devoted to proving Lemma 2.

4 König Graphs with Extendable Vertex Covers

In this section, we obtain a characterization of those König graphs, in which, a given subset of vertices can be extended to a minimum vertex cover. Recall that whenever we say a *minimum vertex cover* of a König graph, we mean a vertex cover that has size equal to the size of a maximum matching. We start off with a couple of definitions.

Definition 2. *Given a graph* $G = (V, E)$ *and a matching* M*, we call a path* $P = v_1, \ldots, v_t$ *in the graph, an* $M-$*alternating path if the edge* $(v_1, v_2) \in M$*, every subsequent alternate edge is in* M *and no other edge of* P *is in* M*. An odd length* $M-$*alternating path is called an odd* $M-$*path and an even length* $M-$*alternating path is called an even* $M-$*path.*

A simple and useful observation to the above definition is the following.

Observation 1. *In odd* $M-$*paths, the last edge of the path is a matched edge, while in even* $M-$*paths, the last edge is not a matching edge.*

Note that, by our definition, a single matching edge is an odd $M-$ path. In addition, we consider a path consisting of a single vertex to be an even $M-$ path by convention. Let $P = v_1, \ldots, v_t$ be an odd (similarly even) $M-$path and let $Q_1, Q_2 \subseteq V(G)$ such that $v_1 \in Q_1$ and $v_t \in Q_2$. Then, we say that P is an odd (similarly even) $M-$path from Q_1 to Q_2.

Definition 3. *Given a graph* G *and a matching* M*, we define an* $M-$*flower as a walk* $W = v_1, \ldots, v_b, v_{b+1} \ldots v_{t-1}, v_t$ *with the following properties.*

- *The vertex* $v_t = v_b$ *and all other vertices of* W *are distinct.*
- *The subpaths* $P_1 = v_1, \ldots, v_b$ *and* $P_2 = v_1, \ldots, v_{t-1}$ *are odd* $M-$*paths from* v_1 *to* v_b *and* v_1 *to* v_{t-1} *respectively.*
- *The odd cycle* $C = v_b, v_b + 1, \ldots, v_t$*, which has length* $t - b$ *and contains exactly* $\lfloor \frac{t-b}{2} \rfloor$ *edges from* M *is called the blossom.*
- *The odd* $M-$*path* P_1 *is called the stem of the flower, the vertex* v_b *is called the base and* v_1 *is called the root (see Fig. 1).*

Given a set $X \subseteq V(G)$*, we say that* G *has an* X $M-$*flower if there is a* $M-$*flower in* G *such that the root is in* X*. The odd* $M-$*path* $v_{b+1}, v_{b+2}, \ldots, v_{t-1}$ *is called a blossom* $M-$*path from* v_{b+1} *to* v_{t-1}*. Blossom* $M-$ *paths are defined only between the two neighbors of the base which lie in the blossom.*

The following consequences follow from the above definitions.

Lemma 3. *Let* $(G = (A, B, E), M, L, R, k)$ *be an instance of* A-AGVC*.*

(a) There cannot be an odd $M-$*path from* A *to* A*.*

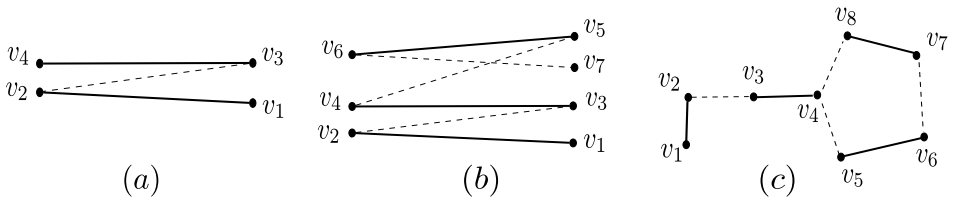

Figure 1: Illustrations of the two types of $M-$alternating paths and an $M-$flower. The non matching edges are represented by the dashed lines. (a) An odd $M-$path v_1, v_2, v_3, v_4. (b) An even $M-$path $v_1, v_2, v_3, v_4, v_5, v_6, v_7$. (c) An $M-$flower with root v_1, base v_4, stem v_1, v_2, v_3, v_4, blossom $v_4, v_5, v_6, v_7, v_8, v_4$, and a blossom path v_5, v_6, v_7, v_8.

(b) Any odd $M-$path from B to B has to contain exactly one edge between two vertices in A.

(c) There cannot be an R $M-$flower with its base in B.

(d) Let $\mathcal{P}$ be an R $M-$flower with base v and let the neighbors of v in the blossom be u_1 and u_2. Then, $u_1 \in B$ or $u_2 \in B$.

(e) Let $P = v_1, \ldots, v_t$ be an odd $M-$path in G and suppose S is a minimum vertex cover of G. If $v_1 \in S$, then $v_t \notin S$.

(f) Let $P = v_1, \ldots, v_t$ be an odd $M-$path from B to B. Then there is an edge (u, v) such that $u, v \in A$ and there is an odd $M-$path P_1 from v_1 to u and an odd $M-$path P_2 from v_t to v and P_1 and P_2 are subpaths of P.

(g) Consider a vertex v and an even $M-$path P from some vertex u to v. Then P does not contain the matching partner of v.

Proof. (a) Any odd $M-$path from A to A will be an odd length path from A to A in the bipartite graph G_{bip} obtained by making A independent, which is not possible.

(b) Any odd $M-$ path $P = v_1, \ldots, v_t$ from B to B which does not contain an edge between two vertices in A, is a path from B to B in the bipartite graph G_{bip}. But an odd $M-$path is by definition an odd length path and such a path cannot exist between two vertices of the same partition in a bipartite graph. Hence it has to use at least one edge between two vertices of A. Now, suppose there are two such edges $e_1 = (v_i, v_{i+1})$ and $e_2 = (v_j, v_{j+1})$ in P and assume without loss of generality that $i+1 < j$. Then, the edges $(v_{i+1}, v_{i+2}), (v_{j-1}, v_j)$ must be in M. They cannot be the same edge since that would mean a matching edge between vertices of A. But now, the subpath $v_{i+1}, v_{i+2}, \ldots, v_j$ an A to A odd $M-$path, which is not possible by (a).

(c) Consider an R $M-$flower and let $b \in B$ be the base. Let u_1 and u_2 be the neighbors of b in the blossom. Since B is independent, $u_1, u_2 \in A$. But, by the definition of flowers, the blossom $M-$path between u_1 and u_2 is an odd $M-$path, which is not possible by (a).

(d) Suppose $u_1, u_2 \in A$. Then the blossom path from u_1 to u_2 is an odd $M-$path from A to A which contradicts (a).

(e) Note that since S is a minimum vertex cover for the König graph G, S must contain exactly one end point of each matched edge in M. We prove by induction on t that if $v_1 \in S$, then $v_t \notin S$. In the base case, $t = 2$. Our claim is clearly true. So, let $t > 2$. Our induction hypothesis assumes that our claim is true $\forall t' < t$. Consider the path $v_1, \ldots, v_{t-2}$. It is clearly an odd $M-$path. By induction hypothesis, $v_1 \in S$ implies $v_{t-2} \notin S$. Since S is a vertex cover, $v_{t-1} \in S$ in order to cover the edge (v_{t-2}, v_{t-1}). Now, for S to be a minimum vertex cover, it can contain exactly one vertex from the edge (v_{t-1}, v_t) since this edge is a matched edge. Hence, $v_t \notin S$ and we have proved our claim.

(f) Let $P = v_1, \ldots, v_t$ be an odd $M-$path from B to B. Let v_i and v_{i+1} be the vertices occuring in P such that $v_i, v_{i+1} \in A$ given by (b). Let P_1 be the subpath of P from v_1 to v_i and let P_2 be the subpath of P from v_{i+1} to v_t. We claim that P_1 and P_2 are both $M-$alternating paths. Note that $v_i, v_{i+1} \in A$ implies that $(v_i, v_{i+1}) \notin M$. Since P is an $M-$alternating path, the edges $e_1 = (v_{i-1}, v_i)$ and $e_2 = (v_{i+1}, v_{i+2})$ are in M. There P_1 is an $M-$alternating path ending in a matching edge, which is precisely the definition of an odd $M-$path. The same is the case for P_2.

(g) Without loss of generality assume that $v = a_i$ for some i. The case where $v = b_j$ for some j is symmetrical. Suppose P contains b_i. Since a_i is the last vertex, and one of the two edges of P incident on b_i is a matched edge, b_i has to occur immediately before a_i, resulting in the last edge of P being a matched edge. But this contradicts our assumption that P is an even $M-$path.

□

Using the above observations, we prove the following characterization.

Lemma 4. *Given an instance $(G = (A, B, E), M, L, R, k)$ of* A-AGVC*, G has a minimum vertex cover S such that $L \cup R \subseteq S$ if and only if there is neither an odd $M-$path from $L \cup R$ to $L \cup R$, nor an R $M-$flower.*

Proof. ($\Rightarrow$) Suppose G has a minimum vertex cover S containing L and R. Recall that $M = \{m_1, \ldots, m_{n/2}\}$ where $m_i = (a_i, b_i)$. Suppose that G has an odd $M-$path $P = v_1, \ldots, v_t$ where $v_1, v_t \in L \cup R \subseteq S$. By Lemma 3(e), it cannot be the case that $v_1, v_t \in S$, a contradiction.

Now, suppose that G has an R $M-$flower $W = v_1, \ldots, v_b, \ldots, v_{t-1}, v_b$ where $v_1 \in R \subseteq S$. By the definition of flowers, the subpath $v_1, \ldots, v_b$ is an odd $M-$path and by Lemma 3(e) $v_b \notin S$. This implies that $v_{b+1}, v_{t-1} \in S$. But, by applying Lemma 3(e) on the blossom $M-$path between v_{b+1} and v_{t-1}, at most one of v_{b+1} and v_{t-1} can be in S, a contradiction.

($\Leftarrow$) Suppose G has no odd $M-$paths from $L \cup R$ to $L \cup R$ and no R $M-$flowers. We define a set S_1 as follows. S_1 contains R and all those vertices to which there is an even $M-$path from R. We then take those matching edges $m_i = (a_i, b_i)$ from which neither end point has yet been picked in S_1 and add the vertex a_i from each such m_i into S_1 and call this new set S. Formally,

$$S_1 = R \cup \{v | \text{ there is an even } M-\text{path from } R \text{ to } v\}.$$

$S = S_1 \cup \{a_i | V(m_i) \cap S_1 = \phi\}$.

We claim that S is a minimum vertex cover of G, containing $L \cup R$. First, we prove that S contains $L \cup R$. By definition, S contains R. Suppose there is a vertex a_i in L which is not in S. By the way we defined S, $b_i \in S_1$ or $b_i \in R$ and hence there is an even $M-$path P from R to b_i. By Lemma 3(g) P does not contain a_i. Now $P + (b_i, a_i)$ is an odd $M-$path from R to L, which is not possible by our assumption. Hence, $L \cup R \subseteq S$.

We now prove that S is indeed a vertex cover. Suppose it is not. Then, let $e = (a_i, b_j)$ be an uncovered edge. Note that by the way we defined S, e cannot be a matching edge because if S_1 picked neither endpoint of a matching edge, we will have added the vertex of that edge lying in A, into S. Since $a_i \notin S$, it must be the case that $b_i \in S_1$. Hence there is an even $M-$path $P = v_1, \ldots, v_t$ from R to b_i where $v_t = b_i$. By Lemma 3(g) P does not contain a_i. First, we assume that P does not contain b_j. In this case, $P + (b_i, a_i) + (a_i, b_j)$ is an even $M-$path from R to b_j which implies that $b_j \in S_1 \subseteq S$, contradicting our assumption that e was uncovered. Hence, we can assume that P contains b_j. Since one of the edges of P incident on b_j is a matched edge, a_j occurs just before or just after b_j in P. If a_j occurs just after b_j, then the subpath of P from v_1 to b_j is an even $M-$path, in which case b_j would have been added to S_1, thus covering the edge e. Hence, a_j must occur just before b_j in P. Let P' be the subpath of P from v_1 to b_j. Now, $P' + (b_j, a_i)$ is an even $M-$path from R to a_i, in which case we would have added a_i to S_1, thus covering e. We have thus established that S is indeed a vertex cover of G.

We now prove that S is a minimum vertex cover of G. In particular we will prove that S contains exactly one vertex from every edge of M. Since we have already shown that S is a vertex cover, it contains at least one vertex from every matching edge. Therefore it is enough for us to prove that S does not pick both the end points of any matching edge. Suppose there is a matching edge m_j such that both a_j and b_j are in S. Then it must be the case that both a_j and b_j are in S_1 as well. This could only be possible if there were even $M-$paths $P_1 = x_1, \ldots, x_s$ and $P_2 = y_1, \ldots, y_t$ from R to a_j and b_j respectively. For each matching edge m_j such that S contains both end points, let P_1^j and P_2^j be the even $M-$paths of least length from R to a_j and b_j respectively. Among all such matching edges, let m_i be one which minimizes $|P_1^i| + |P_2^i|$. In the rest of the proof of this Lemma, we will refer to the paths P_1^i and P_2^i as P_1 and P_2. By Lemma 3(g) P_1 does not contain b_i and P_2 does not contain a_i. We now consider the following two cases.

1. If P_1 and P_2 do not intersect at all, then $P_1 + (a_j, b_j) + Rev(P_2)$ is an odd $M-$path from R to R, which is not possible by our assumption.

2. Suppose P_1 and P_2 do intersect and let $y_{q+1} = x_p$ be the last vertex along P_2 which is also present in P_1 (see Fig. 2). If the edge (y_{q+1}, y_{q+2}) is also contained in P_1, then it contradicts our choice of y_{q+1}. Hence the edge (y_{q+1}, y_{q+2}) cannot be contained in P_1.

   (i) Assume that the edge (y_q, y_{q+1}) is not in P_1 or in other words the edges (y_q, y_{q+1}), (y_{q+1}, y_{q+2}), (x_{p-1}, x_p) and (x_p, x_{p+1}) are distinct. Note that since P_1 and P_2 are both $M-$alternating paths, exactly two of these four distinct edges must be in M. But this would imply that there are two distinct matching edges incident on y_{q+1} which is not possible.

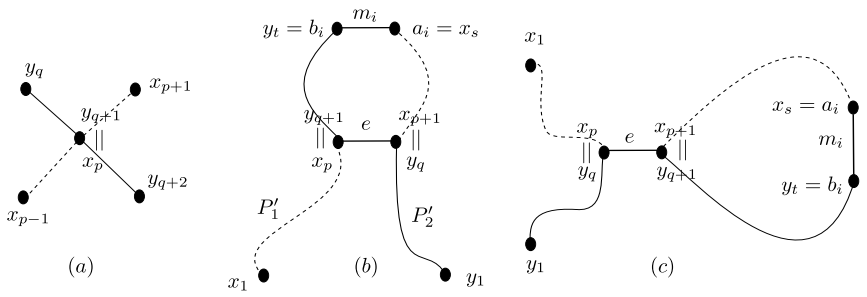

Figure 2: Illustration of the sub cases (i), (ii) and (iii) of case 2 of Lemma 4 with P_1 being the path with dashes and P_2 the one without. (a) Case where P_1 and P_2 do not share an edge adjacent to y_{q+1}. (b) Case where $x_p = y_{q+1}$ and $x_{p+1} = y_q$. (c) Case where $x_p = y_q$ and $x_{p+1} = y_{q+1}$.

(ii) Suppose that P_1 and P_2 share the edge $e = (y_q, y_{q+1}) = (x_p, x_{p+1})$ and $x_p = y_{q+1}$ and $x_{p+1} = y_q$. Note that since b_i cannot occur in P_1 and a_i cannot occur in P_2, $e \neq m_i$. If $e \notin M$, then $(x_{p-1}, x_p), (y_{q+1}, y_{q+2}) \in M$, which is not possible unless they are the same edge. But this contradicts our choice of y_{q+1} as the last vertex along P_2 which is present in P_1. Hence we can assume that $e \in M$. Let P_1' be the subpath of P_1 from x_1 to x_p, and let P_2' be the subpath of P_2 from y_1 to y_q. Since $e \in M$, neither (x_{p-1}, x_p) nor (y_{q-1}, y_q) is in M which implies that both P_1' and P_2' are even $M-$paths from R to x_p and y_q respectively. Since P_1 and P_2 contain at least one more edge than P_1' and P_2' respectively $|P_1'| + |P_2'| < |P_1| + |P_2|$, contradicting our choice of m_i.

(iii) Suppose that P_1 and P_2 share the edge $e = (y_q, y_{q+1}) = (x_p, x_{p+1})$ and $x_p = y_q$ and $x_{p+1} = y_{q+1}$. If $e \notin M$ it forces the edges (x_{p+1}, x_{p+2}) and (y_{q+1}, y_{q+2}) to be matching edges, which is possible only if they are the same edge. But this would contradict our choice of y_{q+1}. Hence, we assume that $e \in M$. Now we have an R $M-$flower with stem $y_1, \ldots, y_{q+1}$, base y_{q+1} and blossom $y_{q+1}, \ldots, b_i + (b_i, a_i) + a_i, x_{s-1}, \ldots, x_{p+1}$. But by our assumption, R flowers do not exist.

This concludes the proof that S is a minimum vertex cover of G containing $L \cup R$. □

We now prove the following simple lemma which handles the base case of the algorithm.

Lemma 5. *Given an instance $(G = (A, B, E), M, L, R, k)$ of* A-AGVC*, if G has a minimum vertex cover containing $L \cup R$, then one such minimum vertex cover can be computed in polynomial time.*

Proof. Since $L \cup R$ is to be a part of the minimum vertex cover, the partner of any vertex in $L \cup R$ cannot be in the vertex cover. Consider the graph obtained by removing the vertices of $L \cup R$ and their partners from G. It is easy to see that this graph is also a König graph and it is sufficient for us to find a minimum vertex cover of this graph and take these vertices along with the vertices of $L \cup R$ to get a minimum vertex cover of G. Finding a minimum vertex cover of a König graph can be done in polynomial time [10] and hence we can compute a minimum vertex cover of G containing $L \cup R$ in polynomial time. □

5 Important Separators and the Algorithm

In this Section we use Lemma 4 to model the A-AGVC problem as a problem of eliminating certain structures in the graph and develop an efficient algorithm for the problem. The overall idea of our algorithm is that we use important separators (defined below) to eliminate odd $M-$paths from $L \cup R$ to $L \cup R$ and when no such path exists, we find an edge and recursively try to solve the problem by including one of the end-points in our potential vertex cover. A well chosen measure allows us to capture this progress and finally lead to the faster algorithm.

5.1 Important Separators in Directed Graphs

The notion of important separators was formally introduced in [16]. Here we extend these definitions to directed graphs in a very natural way. Given a directed graph $D = (V, A)$, consider a set $X \subseteq V$. We denote by $\delta^+_G(X)$, the set of arcs going out of X in D and we define a function $f : 2^V \to \mathbb{N}$ where $f(X) = |\delta^+_G(X)|$. It is easy to verify that the function f is submodular.

Let $X, Y \subset V$ be two disjoint vertex sets. A set $S \subseteq A$ is called an $X - Y$ *separator* or an *arc separator* if no vertex in Y is reachable from any vertex in X in $D \setminus S$. We call S a minimal separator if no proper subset of S is an $X - Y$ separator and denote by $K_{X,S}$ the set of vertices reachable from X in the graph $D \setminus S$. We drop the explicit reference to X if it is clear from the context and just call this set K_S. We let $\lambda_D(X, Y)$ denote the size of the smallest $X - Y$ separator in D. We drop the subscript D when it is clear from the context.

Definition 4. *Let $X, Y \subset V$ be two disjoint vertex sets of D and let $S, S_1 \subseteq V$ be two $X-Y$ separators. We say that S_1 dominates S with respect to X if $|S_1| \leq |S|$ and $K_{X,S} \subset K_{X,S_1}$. We drop the explicit reference to X if it is clear from the context. We call S an important $X - Y$ separator if it is minimal and there is no $X - Y$ separator that dominates S with respect to X.*

Note that, if Y is not reachable from X in D, then the empty set is a trivial important $X-Y$ separator. We make the following observations about important separators, which we will use later in the algorithm.

Lemma 6. *Let $D = (V, A)$ be a directed graph where $|V| = n$, $X, Y \subset V$ be two disjoint vertex sets and S be an important $X - Y$ separator.*

1. *For every $e = (u, v) \in S$, $S \setminus \{e\}$ is an important $X-Y$ separator in the graph $D \setminus \{e\}$.*
2. *If S is an $X' - Y$ arc separator for some $X' \supset X$ such that X' is reachable from X in $D[X']$ where $D[X']$ is the subgraph of D induced on the vertices of X', then S is also an important $X' - Y$ separator.*
3. *There is a unique important $X - Y$ separator S^* of size $\lambda(X, Y)$ and it can be found in polynomial time. Furthermore, $K_{S^*} \subseteq K_S$.*

Proof. 1. Since we are not adding any arcs, $S_1 = S \setminus \{e\}$ is an $X-Y$ separator in $D \setminus \{e\}$. Suppose S_1 is not an important $X-Y$ separator, then there exists an $X-Y$ separator S_2 such that $|S_2| \leq |S_1|$ and $K_{S_2} \supset K_{S_1}$. But then, $S_2 \cup \{e\}$ is an $X - Y$ separator in

G such that $|S_2 \cup \{e\}| \leq |S|$ and $K_{S_2 \cup \{e\}} \supset K_S$ contradicting our assumption that S was an important $X - Y$ separator in G.

2. Observe that it is sufficient for us to prove that any $X' - Y$ separator S_1 which dominates S with respect to X' also dominates S with respect to X. Consider an $X' - Y$ vertex separator S_1 dominating S with respect to X'. We have that $K_{X',S} \subset K_{X',S_1}$. Since X' is reachable from X in $D[X']$ every vertex reachable from X' is also reachable from X in the graphs $D \setminus S$ and $D \setminus S_1$. The only other vertices reachable from X in both these graphs are the precisely the vertices of X' and hence $K_{X,S} \subset K_{X,S_1}$ and hence S_1 dominates S with respect to X.

3. Suppose there are two distinct important separators S_1 and S_2 of size $\lambda(X, Y)$. By the submodularity of f we have that $f(K_{S_1}) + f(K_{S_2}) \geq f(K_{S_1} \cup K_{S_2}) + f(K_{S_1} \cap K_{S_2})$. But $f(K_{S_1}) = f(K_{S_2}) = \lambda(X, Y)$. Observe that $\delta^+(K_{S_1} \cap K_{S_2})$ is also an $X - Y$ separator and hence $f(K_{S_1} \cap K_{S_2}) \geq \lambda(X, Y)$. This implies that $f(K_{S_1} \cup K_{S_2}) \leq \lambda(X, Y)$ which cannot happen unless $S_1 = S_2$.

   We can find the unique important separator of size $\lambda(X, Y)$ as follows. First we find a minimum size $X - Y$ separator S which can be done in polynomial time by standard network flow techniques [1]. We then test if there is an arc $e = (u, v) \in S$ and an $X - Y$ separator of size at most $|S|$ which does not contain any arc in $D[K_s \cup \{e\}]$. This can be done as follows. For each arc $e = (u, v) \in S$, set $X' = K_{S^*} \cup \{v\}$ and find a minimum size $X' - Y$ separator S' in D. We know by Lemma 6(2) that S' is also an $X - Y$ separator. If size of any such S' is at most that of S then S is not important since $K_{S'} \supset K_S$. We can repeat the test with the new separator as S and verify that it is either important or if not get another separator S' such that $|S'| \leq |S|$ and $K_{S'} \supset K_S$. We can repeat this as many times as necessary to find an important $X - Y$ separator of size $\lambda(X, Y)$. We will perform at most n repetitions of each test and the time required for a single test is clearly polynomial in n. Hence we can compute the smallest important separator in polynomial time.

   Finally, we show that K_{S^*} is contained in K_S. Suppose that $K_{S^*} \setminus K_S \neq \phi$. By the submodularity of f we have that $f(K_{S^*}) + f(K_S) \geq f(K_{S^*} \cup K_S) + f(K_{S^*} \cap K_S)$. But $f(K_{S^*}) = \lambda(X, Y)$ and $f(K_{S^*} \cap K_S) \geq \lambda(X, Y)$. This implies that $f(K_S) \geq f(K_{S^*} \cup K_S)$ which contradicts that S was an important separator.

□

5.2 The Algorithm

Note that, given an instance $(G = (A, B, E), M, L, R, k)$ of A-AGVC, in order to find a minimum vertex cover containing $L \cup R$, it is *sufficient* to find the set M' of matched edges which have both end points in this minimum vertex cover. This follows from the fact that the graph $G \setminus V(M')$ is König and has a minimum vertex cover that contains $(L \cup R) \setminus V(M')$. Thus, by Lemma 5, a minimum vertex cover of $G \setminus V(M')$ containing $(L \cup R) \setminus V(M')$ can be computed in polynomial time. Hence, in the rest of the paper whenever we talk about *a solution S for an instance of* A-AGVC*, we will mean the set of edges of M which have both endpoints in the vertex cover.* Given an instance of A-AGVC we define a directed graph $D(G)$

corresponding to this instance as follows. Remove all the edges in $G[A]$, orient all the edges of M from A to B and all the other edges from B to A. An immediate observation to this is the following.

Observation 2. *There is a path from L to R in $D(G)$ if and only if there is an odd $M-$path from L to R in G*

Even though the edges of $D(G)$ are directed (and henceforth will be called arcs), they come from G and have a fixed direction. Hence we will occasionally use the same set of arcs/edges in both the undirected and directed sense. For example we may say that a set S of edges of G is both a solution for the corresponding instance (undirected) and an arc separator in the graph $D(G)$ (directed). The next Lemma characterizes the $L-R$ separators in $D(G)$ and the lemma following it gives an upper bound on the number of such separators.

Lemma 7. *Given an instance $(G=(A,B,E),M,L,R,k)$, any important $L-R$ separator in $D(G)$ comprises precisely arcs corresponding to some subset of M.*

Proof. Let X be an important $L-R$ separator in $D(G)$. Suppose there is an arc $e=(b_j,a_i)\in X$ such that $e\notin M$. The minimality of X implies that b_j and a_i are reachable from L in $D(G)$ but only b_j is reachable from L in $D(G)\setminus X$. Now, consider the set $X'=(X\setminus e)\cup(a_i,b_i)$. Clearly $|X'|\leq|X|$. Now, X' is also a minimal $L-R$ separator in $D(G)$ since any $L-R$ path passing through the arc e also passes through the arc (a_i,b_i). Now, a_i is reachable from L in $D(G)\setminus X'$ and all vertices reachable from L in $D(G)\setminus X$ are also reachable from L in $D(G)\setminus X'$. Hence the set of vertices reachable from L in $D(G)\setminus X'$ is a strict superset of the set of vertices reachable from L in $D(G)\setminus X$. This contradicts our assumption that X was an important $L-R$ separator. □

Lemma 8. *Let $(G=(A,B,E),M,L,R,k)$ be an instance of* A-AGVC *and let $D=D(G)=(V,A)$ be defined as above. Then the number of important $L-R$ separators of size at most k in the graph D is bounded by 4^k and these can be enumerated in time $O^*(4^k)$.*

Proof. Given $D,L,R,k\geq 0$ we define a measure $\mu_a(D,L,R,k)=2k-\lambda_D(L,R)$. We prove by induction on $\mu_a(D,L,R,K)$ that there are at most $2^{\mu_a(D,L,R,k)}$ important $L-R$ separators of size at most k. For the base case, if $2k-\lambda_D(L,R)<k$ then $\lambda_D(L,R)>k$ and hence the number of important $L-R$ separators of size at most k is 0. If $\lambda_D(L,R)=0$, it means that there is no path from L to R and hence the empty set alone is the important $L-R$ separator. Consider $D,L,R,k\geq 0$ such that $\mu_a=\mu_a(D,L,R,k)\geq k,\lambda_D(L,R)>0$ and assume that the statement of the Lemma holds for all D',L',R',k' where $\mu_a(D',L',R',k')<\mu_a$.

By Lemma 6(3) there is a unique important $L-R$ separator S^* of size $\lambda_D(L,R)$. Since we have assumed $\lambda_D(L,R)$ to be positive, S^* is non empty. Consider an arc $e=(u,v)\in S^*$. By Lemma 7, there is some i such that $u=a_i$ and $v=b_i$. Any important $L-R$ separator S either contains e or does not contain e. For any important $L-R$ separator S which contains e, $S\setminus\{e\}$ is an important $L-R$ separator in $D\setminus\{e\}$ by Lemma 6(1). Hence the number of important $L-R$ separators of size at most k in D which contain e, is at most the number of important $L-R$ separators of size at most $k-1$ in $D\setminus\{e\}$. Observe that $\lambda_{D\setminus\{e\}}(L,R)=\lambda_D(L,R)-1$ which implies that $\mu_a(D\setminus\{e\},L,R,k-1)<\mu_a$ and by the induction hypothesis, the number of important $X-Y$ separators of size at most $k-1$ in

$D \setminus \{e\}$ is bounded by $2^{\mu_a - 1}$ which is also a bound on the number of important $L - R$ arc separators of size at most k in D which contain e.

Now let S be an important $L - R$ separator of size at most k which does not contain e. By Lemma 6(3) we know that $K_S \supseteq K_{S^*}$ and by the minimality of S^*, $a_i \in K_{S^*}$ and since $K_S \supseteq K_{S^*}$, a_i is in K_S. But $e \notin S$, which implies that b_i is in K_S which implies that $K_S \supseteq K_{S^*} \cup \{b_i\}$. But by Lemma 7, no other edge incident on b_i can be in S. Hence the vertices in $\delta^+(b_i)$ are also reachable from L in $D \setminus S$. We now set $X = K_{S^*} \cup \delta^+(b_i)$ and by Lemma 6(2) we know that S is also an important $X - R$ separator. We set $L' = A \cap X$. Since there cannot be paths from R to R in $D(G)$, any $X - R$ separator is also an $L' - R$ separator. Thus a bound on the number of important $L' - R$ separators of size at most k is also a bound on the number of important $L - R$ separators of size at most k which do not contain the arc e. First note that $\lambda_D(L', R) > \lambda_D(L, R)$ since otherwise we would have an $L - R$ separator S' of size at most S^* such that $K_{S'} \supset K_{S^*}$. Now, $\mu_a(D, L', R, k) < \mu_a$ and by induction hypothesis, the number of important $L' - R$ separators of size at most k is bounded by $2^{\mu_a - 1}$.

Summing up the upper bounds in both cases we get that the number of important $L - R$ separators of size at most k is bounded by $2.2^{\mu_a - 1} = 2^{\mu_a} \leq 2^{2k}$.

The algorithm for enumerating the important $X - Y$ separators follows from the above proof. The algorithm first computes the unique smallest important $L - R$ separator S^* using the algorithm described in Lemma 6(3), selects an arc $e \in S^*$ and recursively enumerates all important $L - R$ separators which contain e and those which do not. It follows from the above proof that this algorithm runs in time $O^*(4^k)$. □

Lemma 9. *If $(G = (A, B, E), M, L, R, k)$ is a* YES *instance of* A-AGVC*, then it has a solution $\hat{S}$ which contains an important $L - R$ separator in $D(G)$.*

Proof. Let S be a solution for the given instance and let S_{LR} be a minimal subset of S such that in the graph $G \setminus V(S_{LR})$, there is no odd $(M \setminus S_{LR})-$path from $L \setminus V(S_{LR})$ to $R \setminus V(S_{LR})$. Let K be the set of vertices reachable from L in $D(G) \setminus S_{LR}$. If S_{LR} is an important $L - R$ separator, we are done by setting $\hat{S} = S$. Suppose that is not the case. Then, there is an important $L - R$ separator S'_{LR} such that $|S'_{LR}| \leq |S_{LR}|$ and $K' \supset K$ where K' is the set of vertices reachable from L in $D(G) \setminus S'_{LR}$. Consider the set $\hat{S} = (S \setminus S_{LR}) \cup S'_{LR}$. Clearly, $|\hat{S}| \leq |S|$ and $\hat{S}==$ contains an important (L, R) separator in $D(G)$. It remains to prove that $\hat{S} \subseteq M$ and that $\hat{S}$ is indeed a solution to the given instance.

We have proved that $S'_{LR} \subseteq M$ (Lemma 7) and hence $\hat{S} \subseteq M$. Now, suppose $\hat{S}$ is not a solution for this instance. Then $\hat{G} = G \setminus V(\hat{S})$ is a König graph that does not have a minimum vertex cover that contains $(L \cup R) \setminus V(\hat{S})$. Thus by Lemma 4, there is either an odd $M-$path from $L \cup R$ to $L \cup R$ or an R $M-$flower in $\hat{G}$. Since $\hat{S}$ is an $L - R$ separator in $D(G)$, $\hat{G}$ does not have odd $M-$paths from L to R. Hence the only possible obstructions are an M-path from R to R or an R $M-$flower. Note that either of these structures must use an edge in $S'' = S_{LR} \setminus S'_{LR}$. We now consider both these cases separately.

1. Suppose there is an odd $M-$path $P = v_1, \ldots, v_t$ from R to R in $\hat{G}$. Let $e = (v_i, v_{i+1})$ be an edge of S'' on the path P. Since $S'' \subseteq S \subseteq M$, e is a matching edge. If $v_i \in B$ and $v_{i+1} \in A$, we will consider the path $Rev(P)$. Hence, without loss of generality assume that that $v_i \in A$ and $v_{i+1} \in B$. Hence, $v_i \in K$ and $v_{i+1} \notin K$.

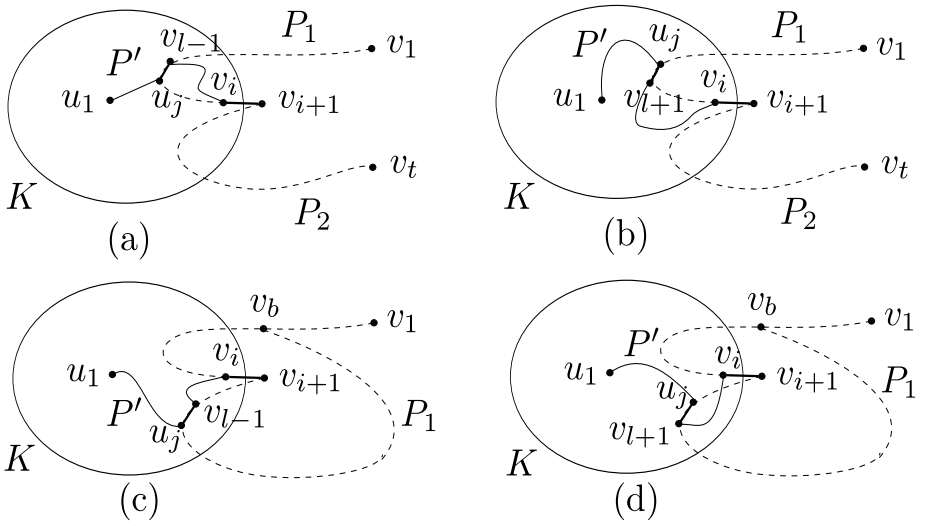

Figure 3: Cases of Lemma 9. (a) Sub case of case 1 where P' intersects P_1 and $(u_j, u_{j+1}) = (v_l, v_{l-1})$. (b) Sub case of 1 where P' intersects P_1 and $(u_j, u_{j+1}) = (v_l, v_{l+1})$ (c) Sub case of 2 where e is part of the blossom, P' intersects P_1 and $(u_j, u_{j+1}) = (v_l, v_{l-1})$ (d) Sub case of 2 where e is part of the blossom, P' intersects P_1 and $(u_j, u_{j+1}) = (v_l, v_{l+1})$

Let P_1 be the subpath of P from v_1 to v_i and let P_2 be the subpath of P from v_{i+1} to v_t. Let $P' = u_1, \ldots, u_p$ be an $M-$alternating path from L to v_i which lies entirely inside K. We know that such a path exists by the minimality of S_{LR}. If P' does not intersect P_1 or P_2, then $P' + (v_i, v_{i+1}) + P_2$ is an odd $M-$path from L to R in $\hat{G}$, which is not possible. Suppose that P' intersects P_1 or P_2 and let $u_j = v_l$ be the first vertex of P' which is present in P_1 or P_2. Please refer to Figure 3 for an illustration of this case. We know that it cannot be the case that these two paths share a vertex but share no edge adjacent to this vertex since that would imply two distinct matching edges incident on the same vertex. We also know that the shared edge cannot lie before u_j in P' since it contradicts our choice of u_j. Hence we know that the edge (u_j, u_{j+1}) is the same as (v_l, v_{l-1}) or (v_l, v_{l+1}) and is a matching edge. In the former case, the path $u_1, \ldots, u_j, v_{l-1}, v_{l-2}, \ldots, v_1$ is an odd $M-$path from L to R in $\hat{G}$ and in the latter case, the path $u_1, \ldots, u_j, v_{l+1}, \ldots, v_t$ is an odd $M-$path from L to R in $\hat{G}$, neither of which is possible.

2. We refer to Figure 3 for an illustration of this case. Suppose there is an R $M-$flower $W = v_1, \ldots, v_b, \ldots, v_{t-1}, v_b$ in $\hat{G}$. Let $e = (v_i, v_{i+1})$ be an edge of S'' on the walk W. We first assume that e is part of the stem of W. Suppose $v_i \in A$ and $v_{i+1} \in B$. Since B is an independent set we have that $v_b \in B$ which contradicts Lemma 3(c). Hence we assume that $v_i \in B$ and $v_{i+1} \in A$ which implies that $v_{i+1} \in K$ and $v_i \notin K$. Let $P' = u_1, \ldots, u_p$ be an $M-$alternating path from L to v_{i+1} which lies entirely inside K. The minimality of S_{LR} guarantees us such a path. Let P_1 be the path $v_1, \ldots, v_i$. If P' does not intersect P_1, then $P' + (v_{i+1}, v_i) + Rev(P_1)$ is an odd $M-$path from L to R which is not possible. Hence assume that P' does intersect P_1 and let u_j be the first vertex of P' which is present in P_1 and let $u_j = v_l$. We know that it cannot be the case that these two paths share a vertex but share no edge adjacent to this vertex since that would imply two distinct matching

edges incident on the same vertex. We also know that the shared edge cannot lie before u_j in P' since it contradicts our choice of u_j. Hence we know that the edge (u_j, u_{j+1}) is the same as (v_l, v_{l-1}) or (v_l, v_{l+1}) and is a matching edge. In the former case, the path $u_1, \ldots, u_j, v_{l-1}, v_{l-2}, \ldots, v_1$ is an odd $M-$path from L to R in $\hat{G}$ and in the later case the path $u_1, \ldots, u_j, v_{l+1}, \ldots, v_i + (v_i, v_{i+1})$ is an odd $M-$path from A to A neither of which is possible, the first because $\hat{G}$ does not have any odd $M-$paths from L to R and the second by Lemma 3(a).

We now assume that e is part of the blossom of W. Without loss of generality assume that that $v_i \in A$ and $v_{i+1} \in B$. Hence, $v_i \in K$ and $v_{i+1} \notin K$.

Let $P' = u_1, \ldots, u_p$ be an $M-$alternating path from L to v_i which lies entirely inside K. We know that such a path exists by the minimality of S_{LR}. Let $P_1 = v_{i+1}, \ldots, v_{t-1}, v_b, v_{b-1}, \ldots, v_1$. If P' does not intersect P_1 then $P' + (v_i, v_{i+1}) + P_1$ is an odd $M-$path from L to R in $\hat{G}$, which is not possible. Suppose that P' intersects P_1 and let $u_j = v_l$ be the first vertex of P' which is present in P_1. Then we know that the edge (u_j, u_{j+1}) is the same as (v_l, v_{l-1}) or (v_l, v_{l+1}) and is a matching edge. In the former case, the path $u_1, \ldots, u_j, v_{l-1}, v_{l-2}, \ldots, v_1$ is an odd $M-$path from L to R in $\hat{G}$ and in the latter case, the path $u_1, \ldots, u_j, v_{l+1}, \ldots, v_t, v_{b-1}, \ldots, v_1$ is an odd $M-$path from L to R in $\hat{G}$, neither of which is possible.

This concludes the proof. □

The next lemma is used to handle the case when the instance does not have odd $M-$paths from L to R.

Lemma 10. *Let $(G = (A, B, E), M, L, R, k)$ be an instance of* A-AGVC *such that there are no odd $M-$paths from L to R in G. If there is either an odd $M-$path P from R to R or an R $M-$flower $\mathcal{P}$, then there is an edge (u, v) such that $u, v \in A \setminus L$ and there is an odd $M-$path from u to R and an odd $M-$path from v to R. Moreover, this edge can be found in polynomial time.*

Proof. Suppose $P = v_1, \ldots, v_t$ is an R to R odd $M-$path. Since $v_1, v_t \in R$, $v_1, v_t \in B$ and by Lemma 3(f) there is an edge (u, v) such that $u, v \in A$ and there are odd $M-$paths from u and v to v_1 and v_t respectively, which are odd $M-$paths from u and v to R.

Suppose $\mathcal{P}$ is an R $M-$flower with root $v_1 \in R$ and base b. Let u_1 and u_2 be the neighbors of b in the blossom. We know by Lemma 3(c) that $b \in A$ and by Lemma 3(d) that at least one of u_1 and u_2 is in B. We first assume that $u_1, u_2 \in B$. Applying Lemma 3(f) on the blossom $M-$ path from u_1 to u_2 (see Fig. 4) we know that there is an edge (u, v) such that $u, v \in A$ and there are odd $M-$paths P_1 and P_2 from u_1 to u and u_2 to v respectively which lie inside the blossom $M-$path. Since P_1 and P_2 lie entirely within this blossom $M-$path, they do not intersect the stem of the flower. We also know by the definition of flowers that there are even $M-$paths P_3 and P_4 from the root to u_1 and u_2 respectively. Hence, $Rev(P_1 + P_3)$ and $Rev(P_2 + P_4)$ are odd $M-$paths from u and v respectively to R.

Now, we assume that exactly one of u_1 and u_2, say u_1 is in A. Then, we claim that there is an odd $M-$path from u_1 to R and one from b to R. By the very definition of flower, there is an odd $M-$path P from the root to the base, and hence $Rev(P)$ is an odd $M-$path from b to R. Let P' be the blossom $M-$path from u_1 to u_2. Observe that $P' + ((u_2, b) + Rev(P)$

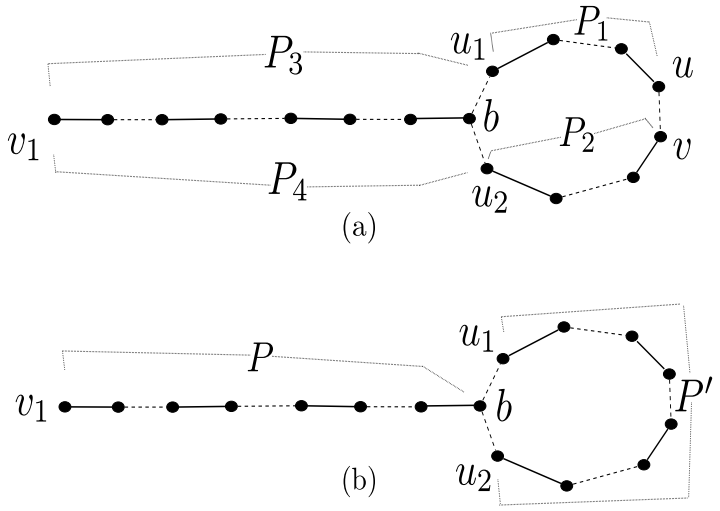

Figure 4: An illustration of the two sub cases for the flower in Lemma 10. (a) $u_1, u_2 \in B$. (b) $u_1 \in A$, $u_2 \in B$.

is indeed an odd $M-$path from u_1 to R. In both these cases, we have found an edge (u, v) such that $u, v \in A$ and there are odd $M-$paths from both these vertices to R. But then, $u, v \in A \setminus L$ since we have assumed that there are no L to R odd $M-$paths.

To find either an R to R odd $M-$path or an R $M-$flower, we proceed by constructing $|R|$ *alternating trees*, one for each vertex of R as follows. We label all the vertices of R as *even*, and keep all the other vertices unlabelled at this point. We repeat the following until either we find an R to R odd $M-$path or an R $M-$flower or all the vertices are labelled. For each even vertex v, label it's matching partner u *odd* and make u a child of v. Now, for each odd vertex u, consider one by one, the edges adjacent to u. The following are the cases which can occur.

1. If there is an edge (u, w) such that w is unlabelled, we label w even and make it a child of u. We then label the matching partner of w odd and make it a child of w.

2. If there is an edge (u, w) such that w is labelled odd and w belongs to another alternating tree, it clearly implies the existence of an odd $M-$path from R to R and the algorithm returns this path.

3. If there is an edge (u, w) such that w is labelled odd and w belongs to the same alternating tree, it clearly implies the existence of an R $M-$flower and the algorithm returns this path.

4. If there is an edge (u, w) such that w is labelled even, then do nothing.

Finally, if all the vertices are labelled, the algorithm returns that there is no R to R odd $M-$path or an R $M-$flower. It is clear that this algorithm runs in polynomial time and finds an odd $M-$path from R or R or an R $M-$flower if either of these structures exist in the graph.

□

Input : An instance (G, M, L, R, k) of A-AGVC

Output: A solution of size at most k for the instance (G, M, L, R, k) if it exists and NO otherwise

1 **if** $k < 0$ **then return** NO
2 *Compute a mimimum size $L - R$ arc separator S in the directed graph $D(G)$*
3 **if** $|S| = 0$ **then**
4 | **if** *there an odd $M-$path from R to R or an R $M-$flower* **then**
5 | | *compute the edge $e = (u, v)$ given by Lemma 10*
6 | | $S_1 \leftarrow Solve - AAGVC(G, M, L \cup \{u\}, R, k)$
7 | | **if** S_1 *is not* NO **then return** S_1
8 | | $S_2 \leftarrow Solve - AAGVC(G, L \cup \{v\}, R, k)$
9 | | **return** S_2
10 | **end**
11 | **else return** ϕ
12 **end**
13 **if** $|S| > k$ **then return** NO
14 **else** *Compute the unique minimum size important $L - R$ separator S^* in $D(G)$ (Lemma 6(c)) and select an arc $e = (w, z) \in S^*$*
15 $S_3 \leftarrow Solve - AAGVC(G \setminus \{e\}, M \setminus \{e\}, L, R, k - 1)$
16 **if** S_3 *is not* NO **then return** $S_3 \cup \{e\}$
17 $S_4 \leftarrow Solve - AAGVC(G, M, A \cap (\delta^+_{D(G)}(z) \cup K_{S^*}), R, k)$
18 **return** S_4

Algorithm 5.1: Algorithm $Solve - AAGVC$ for A-AGVC

We are now ready to prove Lemma 2 by describing an algorithm (Algorithm. 5.1) for A-AGVC. The idea of the algorithm is as follows. If there is an odd $M-$path from L to R in G, then by Lemma 9 we know that if there is a solution, there is one which contains an important $L-R$ separator in $D(G)$. Hence we branch on a choice of an important $L - R$ separator. If there are no odd $M-$paths from L to R, but there is either an odd $M-$path from R to R or an R $M-$flower, we use Lemma 10 to the get an edge (u, v) between two vertices in A and guess the vertex which covers this edge and continue. If neither of these two cases occur, then by Lemma 4 the graph has a minimum vertex cover containing $L \cup R$. Such a minimum vertex cover will not contain both end points of any edge of the perfect matching and hence the algorithm returns the empty set. In order to make the analysis of our algorithm simpler, we embed the algorithm for enumerating important separators (Lemma 8) into our algorithm for A-AGVC.

Correctness. The Correctness of Step 1 is obvious. In Steps 6 and 8 we are merely guessing the vertex which covers the edge (u, v), while Step 11 is correct due to Lemma 4. Step 13 is correct because the size of the minimum $L - R$ separator in $D(G)$ is a lower bound on the solution size. Steps 15 and 17 are part of enumerating the important $L - R$ separators as seen in Lemma 8. Since we have shown in Lemma 9 that if there is a solution, there is one which contains an important $L - R$ separator in $D(G)$, these steps are also correct.

Running Time. In order to analyze the algorithm, we define the search tree $\mathbb{T}(G, M, L, R, k)$

resulting from a call to $Solve - AAGVC(G, M, L, R, k)$ inductively as follows. The tree $\mathbb{T}(G, M, L, R, k)$ is a rooted tree whose root node corresponds to the instance (G, M, L, R, k). If $Solve - AAGVC(G, M, L, R, k)$ does not make a recursive call, then (G, M, L, R, k) is said to be the only node of this tree. If it does make recursive calls, then the children of (G, M, L, R, k) correspond to the instances given as input to the recursive calls made inside the current procedure call. The subtree rooted at a child node (G', M', L', R', k') is the search tree $\mathbb{T}(G', M', L', R', k')$.

Given an instance $I = (G, M, L, R, k)$, we prove by induction on $\mu(I) = 2k - \lambda_{D(G)}(L, R)$ that the number of leaves of the tree $\mathbb{T}(I)$ is bounded by $max\{2^{2\mu(I)}, 1\}$. In the base case, if $\mu(I) < k$, then $\lambda(L, R) > k$ in which case the number of leaves is 1. Assume that $\mu(I) \geq k$ and our claim holds for all instances I' such that $\mu(I') < \mu(I)$.

Suppose $\lambda(L, R) = 0$. In this case, the children I_1 and I_2 of this node correspond to the recursive calls made in Steps 6 and 8. By Lemma 10 there are odd $M-$paths from u to R and from v to R. Hence, $\lambda(L \cup \{u\}, R) > 0$ and $\lambda(L \cup \{v\}, R) > 0$. This implies that $\mu(I_1), \mu(I_2) < \mu(I)$. By the induction hypothesis, the number of leaves in the search trees rooted at I_1 and I_2 are at most $2^{\mu(I_1)}$ and $2^{\mu(I_2)}$ respectively. Hence the number of leaves in the search tree rooted at I is at most $2.2^{\mu(I)-1} = 2^{\mu(I)}$.

Suppose $\lambda(L, R) > 0$. In this case, the children I_1 and I_2 of this node correspond to the recursive calls made in Steps 15 and 17. But in these two cases, as seen in the proof of Lemma 8, $\mu(I_1), \mu(I_2) < \mu(I)$ and hence applying induction hypothesis on the two child nodes and summing up the number of leaves in the sub trees rootes at each, we can bound the number of leaves in the sub tree of I by $2^{\mu(I)}$.

Hence the number of leaves of the search tree $\mathbb{T}$ rooted at the input instance $I = (G, M, L, R, k)$ is $2^{\mu(I)} \leq 2^{2k}$. The time spent at a node I' is bounded by the time required to compute the unique smallest $L - R$ separator in $D(G)$ which is polynomial(Lemma 6). Along any path from the root to a leaf, at any internal node, the size of the set L increases or an edge is removed from the graph. Hence the length of any root to leaf path is at most n^2. Therefore the running time of the algorithm is $O^*(4^k)$. This completes the proof of Lemma 2.

5.3 Consequences

Theorem 1 has some immediate consequences. The first one is regarding the following problem.

König Vertex Deletion

Input:	An undirected graph $G = (V, E)$, positive integer k
Parameter:	k
Question:	Does there exist a set S of at most k vertices of G such that $G \setminus S$ is a König graph?

By [19, Theorem 4], we have the following corollary.

Corollary 1. König Vertex Deletion *can be solved in time* $O^*(9^k)$ *time.*

The following two corollaries are regarding Almost 2 SAT and its variant.

ALMOST 2 SAT

Input:	A 2-CNF formula F, positive integer k
Parameter:	k
Question:	Does there exist a set S_c of at most k clauses of F such that $F \setminus S_c$ is satisfiable?

ALMOST 2 SAT(VARIABLE)

Input:	A 2-CNF formula F, positive integer k
Parameter:	k
Question:	Does there exist a set S_v of at most k variables of F such that $F \setminus S_v$ is satisfiable?

It has been mentioned without proof in [6, Open Problem Posed by M. Fellows] that AGVC and ALMOST 2 SAT are equivalent. However, for the sake of completeness, we give a polynomial time parameter preserving reduction from ALMOST 2 SAT to AGVC and hence prove the following Lemma.

Lemma 11. ALMOST 2 SAT *can be solved in* $O^*(9^k)$ *time.*

Proof. The proof is by a parameter preserving polynomial time reduction from ALMOST 2 SAT to AGVC. The reduction is as follows. Let $(F = C_1 \wedge \ldots, \wedge C_m, k)$ be an instance of ALMOST 2 SAT where F is a 2-CNF on n variables $x_1 \ldots, x_n$. We define the instance of AGVC as follows. We construct a graph on $2(m + (k+1)n)$ vertices. We have $2k+2$ vertices $x^1_{i_1}, \ldots, x^1_{i_{k+1}}, x^0_{i_1}, \ldots, x^0_{i_{k+1}}$ corresponding to every variable x_i and two vertices y^1_ℓ and y^0_ℓ corresponding to every clause y_ℓ. We add edges between $x^1_{i_j}$ and $x^0_{i_j}$ for every x_i and every j, and edges between y^1_ℓ and y^0_ℓ for every clause y_ℓ resulting in a perfect matching M for the graph. Now we add edges corresponding to each clause as follows. Consider the $\ell^t h$ clause $C_\ell = (l_1 \vee l_2)$. If $l_1 = \bar{x}_p$ for some p, then add edges from $x^0_{p_j}$ to y^1_ℓ for every $1 \leq j \leq k+1$ and if $l_1 = x_p$ for some p then add edges from $x^1_{p_j}$ to y^1_ℓ for every $1 \leq j \leq k+1$. Similarly if $l_2 = \bar{x}_p$ for some p, then add edges from $x^0_{p_j}$ to y^0_ℓ for every $1 \leq j \leq k+1$ and if $l_2 = x_p$ for some p then add edges from $x^1_{p_j}$ to y^0_ℓ for every $1 \leq j \leq k+1$. We do this for every clause and this concludes the construction of the graph G. We refer to Figure 5 for an illustration of the construction. We claim that (F, k) is a YES instance if and only if (G, M, k) is a YES instance.

Suppose that (F, k) is a YES instance and let S_C be the set of at most k clauses the removal of which makes F satisfiable. Let $S_C = \{C_{\ell_1}, \ldots, C_{\ell_t}\}$ and let P be a satisfying assignment of $F \setminus S$. We now construct a vertex cover S of G as follows. For every $C_{\ell_j} \in S_C$ add the vertices $y^0_{\ell_j}, y^1_{\ell_j}$ to S. From every matching edge $(x^1_{i_j}, x^0_{i_j})$ add $x^1_{i_j}$ to S if P assigns 1 to x_i and add $x^0_{i_j}$ to S if P assigns 0 to x_i. The remaining uncovered matching edges of the form (y^1_ℓ, y^0_ℓ) correspond to undeleted clauses. We add y^1_ℓ to S if some neighbor of y^1_ℓ is not in S and add y^0_ℓ to S otherwise. We claim that S is a vertex cover of G which picks both endpoints of at most k of the matching edges. The second property is clearly satisfied by the way we defined S. It remains to prove that S is a vertex cover. Again by the definition of S, it covers all the matching edges. Suppose that S is not a vertex cover and consider some uncovered edge $e = (x^a_{i_j}, y^b_\ell)$ where $a, b \in \{0, 1\}$ and consider the clause corresponding to y_ℓ,

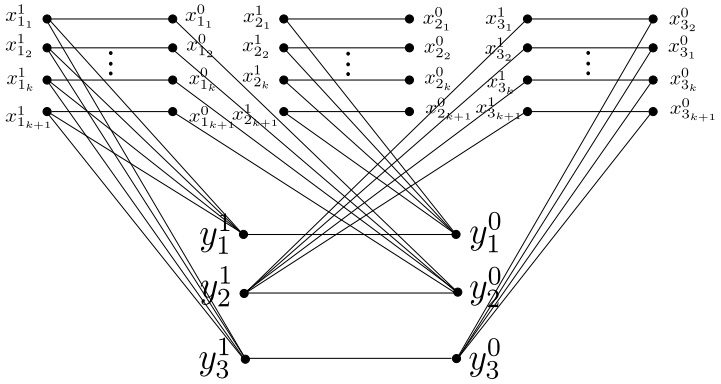

Figure 5: Example for the reduction where $F = (x_1 \vee x_2) \wedge (x_3 \vee \bar{x}_1) \wedge (x_1 \vee \bar{x}_3)$.

$(l_1 \vee l_2)$. As a neighbor of y_ℓ^b is not in S, by construction of S, it must be the case that b is 0. And since $x_{i,j}^a$ is not in S, the literal l_2 involves the variable x_i and is not satisfying the clause y_ℓ. But, since the vertex y_ℓ^0 was not picked in S, it must have been the case that some vertex adjacent to y_ℓ^1 was not picked in S. This means that the literal l_1 is also not satisfying the clause y_ℓ and hence the clause $(l_1 \vee l_2)$ is unsatisfied, and hence must have been in S_C and hence we must have picked both vertices y_ℓ^0 and y_ℓ^1 in S, a contradiction to the fact that e was uncovered by S.

Conversely suppose that (G, M, k) is a YES instance and set S be a vertex cover of G which picks both the end points of at most k matching edges. Observe that the way we have constructed G allows us to assume that the matching edges from which both end points are in S are of the form (y_ℓ^1, y_ℓ^0) and that for every x_i, either $x_{i_j}^1$ is in S for every j or $x_{i_j}^0$ is in S for every j. We let S_C be the set of those clauses corresponding to the matching edges whose both end points are in S. We now define an assignment P for the variables of F as follows. For each x_i set $x_i = 1$ if for every j, $x_{i_j}^1 \in S$ and set $x_i = 0$ if for every j $x_{i_j}^0 \in S$. We claim that P is a satisfying assignment for $F \setminus S_C$. Suppose it is not, and consider a clause C in $F \setminus S_C$ and let $C = C_\ell = l_1 \vee l_2$. Since C_ℓ is unsatisfied, P has assigned 0 to both the literals l_1 and l_2. This implies that the edges corresponding to this clause are covered by the vertices y_ℓ^1 and y_ℓ^0 in which case we would have deleted the clause C. □

By [18, Theorem 3.1], we have the following Corollary.

Corollary 2. ALMOST 2 SAT(VARIABLE) *can be solved in time* $O^*(9^k)$ *time.*

6 Conclusion and Subsequent Research

In this paper, we obtained a faster FPT algorithm for the ABOVE GUARANTEE VERTEX COVER problem . We gave a structural characterization of König graphs in which a minimum vertex cover contains certain fixed vertices. This characterization, given in terms of classical

graph theoretic structures like flowers, allowed us to model a variant of AGVC as a problem of eliminating some of these structures. This allowed us to apply the concept of important separators to solve this problem, and finally, using this algorithm as a subroutine, led to the FPT algorithm for AGVC . This algorithm for AGVC also led to faster FPT algorithms for ALMOST 2 SAT, ALMOST 2 SAT(VARIABLE) and KÖNIG VERTEX DELETION.

Subsequent to our research, several new and exciting developments have happened around AGVC, or equivalently around ALMOST 2 SAT, using entirely different approaches. Cygan et al. [5] obtained an algorithm for an above guarantee version of MULTIWAY CUT running in time $O^*(4^k)$ using a novel branching guided by a solution to the relaxation of the natural linear program for this problem. Then, via a reduction to this problem Cygan et al. obtained an algorithm for AGVC and ALMOST 2 SAT (and other related problems) running in time $O^*(4^k)$. Even more recently, Narayanaswamy et al. [20] obtained an algorithm for AGVC running in time $O^*(2.618^k)$ using a refined branching based on linear programming, and their bound has been improved to $O^*(2.32^k)$ by Lokshtanov et al. [13]. Finally, Kratsch and Wahlström [12] have studied the kernelization complexity of AGVC and obtained a randomized polynomial sized kernel for it through matroid based techniques.

References

[1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network flows*, Prentice Hall Inc., 1993. Theory, algorithms, and applications.

[2] J. CHEN, I. A. KANJ, AND G. XIA, *Improved upper bounds for vertex cover*, Theor. Comput. Sci., 411 (2010), pp. 3736–3756.

[3] J. CHEN, Y. LIU, AND S. LU, *An improved parameterized algorithm for the minimum node multiway cut problem*, Algorithmica, 55 (2009), pp. 1–13.

[4] J. CHEN, Y. LIU, S. LU, B. O'SULLIVAN, AND I. RAZGON, *A fixed-parameter algorithm for the directed feedback vertex set problem*, J. ACM, 55 (2008).

[5] M. CYGAN, M. PILIPCZUK, M. PILIPCZUK, AND J. O. WOJTASZCZYK, *On multiway cut parameterized above lower bounds*, to apper in IPEC, (2011).

[6] E. DEMAINE, G. GUTIN, D. MARX, AND U. STEGE, *Open problems from dagstuhl seminar 07281*, 2007.

[7] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[8] J. EDMONDS, *Paths, trees, and flowers*, Canad. J. Math., 17 (1965), pp. 449–467.

[9] J. FLUM AND M. GROHE, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.

[10] F. GAVRIL, *Testing for equality between maximum matching and minimum node covering*, Inf. Process. Lett., 6 (1977), pp. 199–202.

[11] S. Khot and V. Raman, *Parameterized complexity of finding subgraphs with hereditary properties*, Theor. Comput. Sci., 289 (2002), pp. 997–1008.

[12] S. Kratsch and M. Wahlström, *Representative sets and irrelevant vertices: New tools for kernelization*, CoRR, abs/1111.2195 (2011).

[13] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *Faster parameterized algorithms using linear programming*, Arxiv e-prints, (2012).

[14] L. Lovász and M. D. Plummer, *Matching Theory*, North Holland, Oxford, 1986.

[15] M. Mahajan and V. Raman, *Parameterizing above guaranteed values: Max-Sat and Max-Cut*, J. Algorithms, 31 (1999), pp. 335–354.

[16] D. Marx, *Parameterized graph separation problems*, Theoret. Comput. Sci., 351 (2006), pp. 394–406.

[17] D. Marx and I. Razgon, *Constant ratio fixed-parameter approximation of the edge multicut problem*, Inf. Process. Lett., 109 (2009), pp. 1161–1166.

[18] ——, *Fixed-parameter tractability of multicut parameterized by the size of the cutset*, in STOC, 2011, pp. 469–478.

[19] S. Mishra, V. Raman, S. Saurabh, S. Sikdar, and C. R. Subramanian, *The complexity of König subgraph problems and above-guarantee vertex cover*, Algorithmica, 61 (2011), pp. 857–881.

[20] N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *Lp can be a cure for parameterized problems*, in STACS, 2012, pp. 338–349.

[21] I. Razgon and B. O'Sullivan, *Almost 2-SAT is fixed-parameter tractable.*, J. Comput. Syst. Sci., 75 (2009), pp. 435–450.

[22] B. A. Reed, K. Smith, and A. Vetta, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004), pp. 299–301.