On the MAX k-VERTEX COVER problem

Federico Della Croce, Vangelis Paschos

UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7243

CAHIER DU LAMSADE

307

Mars 2011

On the MAX *k*-VERTEX COVER problem

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Federico Della Croce, Vangelis Th. Paschos

On the MAX k-VERTEX COVER problem*

Federico Della Croce† Vangelis Th. Paschos‡§

Abstract

Given a graph $G(V, E)$ of order n and a constant $k \leqslant n$, the MAX k-VERTEX COVER problem consists of determining k vertices that cover the maximum number of edges in G. In its (standard) parameterized version, MAX k-VERTEX COVER can be stated as follows: "given G, k and parameter ℓ, does G contain k vertices that cover at least ℓ edges?". We first devise moderately exponential exact algorithms for MAX k-VERTEX COVER, with complexity exponential to n (note that the known results concerned time bounds of the form $n^{O(k)}$) by developing a branch and reduce method based upon the measure-and-conquer technique. We then prove that, interestingly enough, although MAX k-VERTEX COVER is non fixed parameter tractable with respect to ℓ, it is fixed parameter tractable with respect to the size τ of a minimum vertex cover of G. We also point out that the same happens for a lot of well-known problems quite different from MAX k-VERTEX COVER. We finally study approximation of MAX k-VERTEX COVER by moderately exponential algorithms. The general goal of the issue of moderately exponential approximation is to catch-up on polynomial inapproximability, by providing algorithms achieving, with worst-case running times importantly smaller than those needed for exact computation, approximation ratios unachievable in polynomial time.

1 Introduction

In the MAX k-VERTEX COVER problem a graph $G(V, E)$ with $|V| = n$ vertices $1, \ldots, n$ and $|E|$ edges (i, j) is given together with an integer value $k < n$. The goal is to find a subset $K \subset V$ with cardinality k, that is $|K| = k$, such that the total number of edges covered by K is maximized. In its (standard) parameterized version, MAX k-VERTEX COVER can be defined as follows: "given G, k and parameter ℓ, does G contain k vertices that cover at least ℓ edges?". MAX k-VERTEX COVER is **NP**-hard (it contains the minimum vertex cover problem as particular case) and has been shown to be **W[1]**-hard in [8] (see [14] for information on fixed parameter (in)tractability). This implies that there is very little hope that an algorithm solving MAX k-VERTEX COVER with running time $O(p(n) \times c^{\ell})$, where $p(n)$ is a polynomial of n and c a fixed constant, could ever be devised.

In what follows, since there exist many natural parameters for a given problem (e.g., for a graph-problem, the value of an optimal solution, the size of a minimum vertex cover of the input-graph, the pathwidth or the treewidth of the input-graph, ..., [14, 17]), we will use notation **FPT(p)** to denote that the fixed parameter tractability considered is with respect to parameter p. In other words, the result by [8], just mentioned can be stated as MAX k-VERTEX COVER $\notin$ **FPT(ℓ)**, unless **FPT(ℓ) = W[1]**. Let us note that, except for the parameters stated just above, MAX k-VERTEX COVER admits another very natural parameter, the integer k.

MAX k-VERTEX COVER is known to be polynomially approximable within approximation ratio $3/4$, while it cannot be solved by a polynomial time approximation schema unless **P = NP**. The interested reader can be referred to [16, 23] for more information about approximation issues for this problem.

In this paper we first consider solution of MAX k-VERTEX COVER by exact and fixed parameter algorithms. Throughout the paper we use notation $O^*(\cdot)$ to measure the running time of an algorithm ignoring polynomial factors. To the authors knowledge, the only available worst-case complexity result for the exact solution of the MAX k-VERTEX COVER is the one of [8], where an exact algorithm with complexity $O^*(n^{\omega\lceil k/3\rceil+O(1)})$ was proposed based upon a generalization of the $O^*(n^{\omega t})$ algorithm of [24] for finding a $3t$-clique in a graph, where $\omega = 2.376$. This induces a complexity $O^*(n^{0.792k})$. Let us note

*Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010

†D.A.I., Politecnico di Torino, federico.dellacroce@polito.it

‡LAMSADE, CNRS UMR 7243 and Université Paris-Dauphine, paschos@lamsade.dauphine.fr

§Institut Universitaire de France

that a trivial optimal algorithm for MAX k-VERTEX COVER takes time $O^*(\binom{n}{k}) = O^*(n^k)$ producing all the subsets of V of size k. This turns to a worst-case $O^*(2^n)$ time (since $\binom{n}{k} \leqslant 2^n$ with equality for $k = \frac{n}{2}$) and no exact algorithm with running time $O^*(\gamma^n)$, for some $\gamma < 2$ seems to be known.

We first explore the possibility of solving MAX k-VERTEX COVER in time better then $O^*(2^n)$. Based upon them, an exact branch and reduce algorithm based upon the measure-and-conquer paradigm by [19] is proposed in Section 2.1 requiring running time $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$, where Δ denotes the maximum degree of G. Such an algorithm is then tailored to graphs with maximum degree 3 inducing a running time $O^*(1.3339^n)$ (Section 2.2). In Section 3) we study fixed parameter tractability of MAX k-VERTEX COVER with respect to the size τ of a minimum vertex cover of G and prove that (although not in $\mathbf{FPT(\ell)}$, the problem is in $\mathbf{FPT(\tau)}$, by devising an algorithm solving it in time $O^*(2^\tau)$. By elaborating a bit more this result we then show that any instance of MAX k-VERTEX COVER is either in $\mathbf{FPT(k)}$, or it can be solved in time c^τ, for $c < 2$. Finally, we show that the technique used for proving that MAX k-VERTEX COVER $\in \mathbf{FPT(\tau)}$ can be used to prove inclusion in the same class of many other well-known combinatorial problems. In Section 4 we study complexity of MAX k-VERTEX COVER in bipartite graphs and prove that it is **NP**-hard in this class. To our knowledge, complexity of the problem in this class has not been studied yet. A corollary of the inclusion of MAX k-VERTEX COVER in $\mathbf{FPT(\tau)}$ is that MAX k-VERTEX COVER in bipartite graphs can be solved in time $O^*(2^{n/2}) \simeq O^*(1.414^n)$. Finally, in Section 5, we address the question of approximating MAX k-VERTEX COVER within ratios "prohibited" for polynomial time algorithms, by algorithms running with moderately exponential complexity. The general goal of this issue is to catch-up on polynomial inapproximability, by developing algorithms achieving, with worst-case running times significantly lower than those needed for exact computation, approximation ratios unachievable in polynomial time. This approach has already been considered for several other paradigmatic problems such as MINIMUM SET COVER [6, 12], MIN COLORING [2, 5], MAX INDEPENDENT SET and MIN VERTEX COVER [4], MIN BANDWIDTH [13, 20], ... Similar issues arise in the field of FPT algorithms, where approximation notions have been introduced, for instance, in [9, 15].

2 Measure-and-conquer and MAX k-VERTEX COVER

2.1 An $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$ algorithm in general graphs

In what follows, we denote by α_j the total number of vertices adjacent to j that have been discarded in the previous levels of the search tree. We denote by d_j the degree of vertex j and by $N(j)$ the set of vertices adjacent to j, that is the neighborhood of j. Notice that, whenever a branch on a vertex j occurs, for each $l \in N(j)$, if j is selected then d_l is decreased by one unit as edge (j, l) is already covered by j. Alternatively, j is discarded: correspondingly d_l is not modified and α_l is increased by one unit. We propose in this section a branch and reduce approach based on the measure-and-conquer paradigm (see for instance [19]). Consider a classical binary branching scheme on some vertex j where j is either selected or discarded. Contrarily to the classical branch-and-reduce paradigm where for each level of the search tree we define as *fixed* those vertices that have already been selected or discarded, while we define as *free* the other vertices, when using measure-and-conquer, we do not count in the measure the fixed vertices, namely the vertices that have been either selected or discarded at an earlier stage of the search tree and we count with a weight w_h the free vertices h. The vertex j to be selected is the one with largest coefficient $c_j = d_j - \alpha_j$. Let $c_{\max}$ denote such a coefficient, hence $c_{\max} \leqslant \Delta$. Then, each free vertex h is assigned a weight $w_h = w_{[i]}$ with $i = c_i = d_h - \alpha_h$ and we impose $w_{[0]} \leqslant w_{[1]} \leqslant w_{[2]} \leqslant w_{[3]} \leqslant \ldots \leqslant w_{[c_{\max}]} = 1$ that is the weights of the vertices are strictly increasing in their c_j coefficients.

We so get recurrences on the time $T(p)$ required to solve instances of size p, where the size of an instance is the sum of the weights of its vertices. Since initially $p = n$, the overall running time is expressed as a function of n. This is valid since when $p = 0$, there are only vertices with weight w_0 in the graph and, in this case, the problem is immediately solved by selecting the $k - \gamma$ vertices with largest α_j (if $\gamma < k$ vertices have been selected so far). Correspondingly free vertices j with no adjacent free vertices receive weight $w_0 = 0$.

We claim that MAX k-VERTEX COVER can be solved with running time $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$ by the following algorithm called `MAXKVC`:

> Select j such that c_j is maximum and branch according to the following exhaustive cases:
>
> 1. if $c_j \geqslant 3$, then branch on j and either select or discard j;

2. else, $c_j \leqslant 2$ and `MAXKVC` is polynomially solvable.

Theorem 1. *Algorithm* ***MAXKVC*** *solves* MAX k-VERTEX COVER *with running time* $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$.

Proof. To prove the above statement, we first show that the branch in step 1 can be solved with complexity $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$ and then we show that step 2 is polynomially solvable. Consider step 1. We always branch on the vertex j with largest $c_j = c_{\max} \leqslant \Delta$ where $c_j \geqslant 3$ and either we select or discard j. If we select j, vertex j is fixed and $c_{\max}$ vertices (the neighbors of j) decrease their degree (and correspondingly their coefficient) by one unit. Similarly, if we discard j, vertex j is fixed and $c_{\max}$ vertices (the neighbors of j) decrease their coefficient as their degree remains unchanged but their α parameter is increased by one unit. Hence, the recurrence becomes:

$$T(p) \leqslant 2T\left(p - w_{[c_{\max}]} - \sum_{h \in N(j)} \left(w_{[c_h]} - w_{[c_h-1]}\right)\right)$$

By constraining the weights to satisfy the inequality $w_{[j]} - w_{[j-1]} \leqslant w_{[j-1]} - w_{[j-2]}$, $\forall j = 2, \ldots, c_{\max}$, the previous recurrence becomes in the worst-case $T(p) \leqslant 2T\left(p - w_{[c_{\max}]} - c_{\max}\left(w_{[c_{\max}]} - w_{[c_{\max}-1]}\right)\right)$. As $c_{\max} \leqslant \Delta$, where the equality occurs when $\alpha_j = 0$, the above recurrence becomes, in the worst-case, $T(p) \leqslant 2T\left(p - w_{[\Delta]} - \Delta\left(w_{[\Delta]} - w_{[\Delta-1]}\right)\right)$.

Summarizing, to handle graphs with maximum degree Δ, we need to guarantee that the recurrences $T(p) \leqslant 2T(p - w_{[i]} - i(w_{[i]} - w_{[i-1]}))$, $\forall i \in 3, \ldots, \Delta$ (as $c_j \geqslant 3$), and the constraints:

$$\begin{array}{rcl} w_{[i]} - w_{[i-1]} & \leqslant & w_{[i-1]} - w_{[i-2]} \ \ \forall i = 2, \ldots, \Delta \\ 0 = w_{[0]} & \leqslant & w_{[1]} \leqslant w_{[2]} \leqslant w_{[3]} \leqslant \ldots \leqslant w_{[\Delta-1]} \leqslant w_{[\Delta]} = 1 \end{array}$$

are satisfied contemporaneously. This corresponds to a non linear optimization problem of the form:

$$\begin{array}{ll} \min & \alpha \\ & \alpha^{(w_{[i]} + i(w_{[i]} - w_{[i-1]}))} \geqslant 2 \ \ \forall i = 3, \ldots, \Delta \qquad (1) \\ & w_{[i]} - w_{[i-1]} \leqslant w_{[i-1]} - w_{[i-2]} \ \ \forall i = 2, \ldots, \Delta \qquad (2) \\ & 0 = w_{[0]} \leqslant w_{[1]} \leqslant w_{[2]} \leqslant w_{[3]} \leqslant \ldots \leqslant w_{[\Delta-1]} \leqslant w_{[\Delta]} = 1 \qquad (3) \end{array}$$

By means of a non linear solver [1], we get Table 1 presenting the corresponding performances of the algorithm for small values of Δ.

Δ	3	4	5	6	7	8	9	10
Complexity	1.4142^n	1.5157^n	1.5874^n	1.6407^n	1.6818^n	1.7145^n	1.7411^n	1.7632^n

Table 1: Complexity results for small values of Δ with the measure-and-conquer approach.

Interestingly enough, for all these values of Δ, the complexity corresponds to $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$. Indeed, this is not accidental. By setting:

$$\begin{array}{rcll} w_{[i]} & = & \dfrac{(i-1)(\Delta+1)}{(i+1)(\Delta-1)} \ \ \forall i = 2, \ldots, \Delta & (4) \\ w_{[1]} & = & \dfrac{1}{2} w_{[2]} & (5) \\ w_{[0]} & = & 0 & (6) \end{array}$$

we can see that constraints (2) and (3) are satisfied. To see that inequalities (2) are satisfied, notice that:

$$\begin{array}{rcl} w_{[3]} - w_{[2]} = w_{[2]} - w_{[1]} & = & \dfrac{1}{3} w_3 \\ w_{[2]} - w_{[1]} = w_{[1]} - w_{[0]} & = & w_1 \end{array}$$

For the general recursion with $i \geqslant 4$, we have to show that $w_{[i]} - w_{[i-1]} \leqslant w_{[i-1]} - w_{[i-2]}$, i.e., that $w_{[i]} - 2w_{[i-1]} + w_{[i-2]} \leqslant 0$. This corresponds to:

$$\begin{aligned}
& \left(\frac{i-1}{i+1} - 2\frac{i-2}{i} + \frac{i-3}{i-1}\right)\left(\frac{\Delta+1}{\Delta-1}\right) \leqslant 0 \implies \frac{i-1}{i+1} - 2\frac{i-2}{i} + \frac{i-3}{i-1} \leqslant 0 \\
\iff\ & i(i-1)^2 - 2(i-2)\left(i^2-1\right) + i(i-3)i + 1 \leqslant 0 \\
\iff\ & i^3 - 2i^2 + i - 2i^3 + 4i^2 + 2i - 4 + i^3 - 2i^2 - 3i = -4 \leqslant 0, \quad \forall i
\end{aligned}$$

Also, to see that inequalities (3) are satisfied, notice that equations (4) imply:

$$\begin{aligned}
w_{[\Delta]} &= 1 \\
w_{[i]} &> 0 \ \ \forall i = 2, \ldots, \Delta \\
w_{[i]} &> w_{[i-1]} \ \ \forall i = 3, \ldots, \Delta
\end{aligned}$$

while equations (5) and (6) imply $w_{[2]} > w_{[1]} > w_{[0]} = 0$.

Finally, notice that such values of $w_{[j]}s$ satisfy constraints (1) that now correspond to $\Delta - 2$ copies of the inequality $\alpha^{\frac{\Delta+1}{\Delta-1}} \geqslant 2$ where the minimum value of α is obviously given by $2^{\frac{\Delta-1}{\Delta+1}n}$. Consequently, the overall complexity of step 1 is $O^*(2^{\frac{\Delta-1}{\Delta+1}n})$.

We consider now step 2. For $c_j = c_{\max} \leqslant 2$, MAX k-VERTEX COVER can be seen as a maximum weighted k-vertex cover problem in an undirected graph G where each vertex j has a weight α_j and a degree $d_j = c_j$ and the maximum vertex degree is 2. But this problem has been shown to be solvable in $O(n)$ time by dynamic programming in [25]. ∎

We now show how the result of Theorem 1 can be improved by *trading space for time*. For this, we use the principle of *memorization* (see [18]) works as follows. Before running the algorithm on the main graph, we run it on every induced subgraph of size at most αn and store the results in a table. In absence of weights, as fixed vertices never change their status, the number of subproblems of size αn to consider is $\binom{n}{\alpha n}$. Besides, thanks to the recurrence defined above, we only need to call a finite number of subproblems of size $k-1$ or less in order to compute a given subproblem of size k. Then, using a classical bottom up technique, the total computation time (and space) for all the subproblems of size k, say $S(k,G)$, is at most $O^*(\binom{n}{k} + \sum_{i \leqslant k-1} S(i,G))$, from what we get (for $\alpha \leqslant 1/2$): $\sum_{k \leqslant \alpha n} S(k,G) \in O^*(\binom{n}{\alpha n})$.

Then, if the polynomial space algorithm has complexity $O^*(\beta)^n$, we run the main algorithm until the remaining graph has size αn or less; then a polynomial-time query in the storage table allows us to conclude. The total running time and space is then $O^*(\max\{\binom{n}{\alpha n}, \beta^{(1-\alpha)n}\})$. This value is then minimal for an α solution of the equation:

$$\beta^{1-\alpha} = \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \tag{7}$$

In our case, due to the presence of weights, we know that to reach a subgraph of size at most αn we need to run the main algorithm until the remaining graph has weight $w_1 \alpha n$ (as in the worst-case all vertices but a constant number may have the smallest weight w_1). Correspondingly, (7) becomes:

$$\beta^{1-w_1\alpha} = \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \tag{8}$$

and, as $\beta = 2^{\frac{\Delta-1}{\Delta+1}}$ and $w_1 = \frac{\Delta+1}{6(\Delta-1)}$, we get $2^{(\frac{\Delta-1}{\Delta+1} - \frac{\alpha}{6})} = \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}}$.

In Table 2, are indicated the complexity results on small values of Δ when the memorization approach is applied.

Δ	3	4	5	6	7	8	9	10
Complexity	1.3973	1.4919	1.5579	1.6066	1.6438	1.6733	1.6971	1.7168

Table 2: Complexity results for small values of Δ with the memorization approach.

2.2 Tailoring measure-and-conquer to graphs with maximum degree 3

We consider now the case where $\Delta = 3$. In order to tailor the measure-and-conquer approach to graphs with $\Delta = 3$, the following remark holds.

Remark 1. The graph can be cubic just once. When branching on a vertex j of maximum degree 3, we can always assume that it is adjacent to at least one vertex k that has already been selected or discarded. That is, either $d_k \leqslant 2$, or $\alpha_k \geqslant 1$, that is $c_k \leqslant 2$. Indeed, the situation where the graph is 3-regular occurs at most once (even in case of disconnection). Thus, we make only one "bad" branching (where every free vertex of maximum degree 3 is adjacent only to free vertices of degree 3). Such a branching may increase the global running time only by a constant factor. ∎

Lemma 1. *Any vertex i with $d_i \leqslant 1$ and $\alpha_i = 0$ can be discarded wlog.*

Proof. If $d_i = \alpha_i = 0$, then i can be obviously discarded. If $d_i = 1$ and $\alpha_i = 0$, then i is adjacent to another free vertex k. But then, if k is selected, i becomes of degree 0 and can be discarded. Alternatively, k is discarded, but then any solution with i but not k is dominated by the one including k instead of i. ∎

Lemma 2. *Any vertex i with $\alpha_i \geqslant 2$ and $d_i = 3$ can be selected wlog.*

Proof. If $\alpha_i = 3$, then i can be obviously selected. If $d_i = 3$ and $\alpha_i = 2$, then i is adjacent to another free vertex k. But then, if k is discarded, we have $\alpha_i = 3$ and i can be selected. Alternatively k is selected, but then any solution with k but not i is dominated by the one including i instead of k. ∎

Then, we claim that we can solve MAX k-VERTEX COVER on graphs with $\Delta = 3$ with running time $O^*(1.3339^n)$ by the following algorithm called `MAXKVC-3`:

Select j such that c_j is maximum and branch according to the following exhaustive cases:

1. if $c_j = 3$, where wlog j is adjacent to i, l, m free vertices with $c_i \leqslant 2$ (due to Remark 1) and $c_i \leqslant c_l \leqslant c_m$, then branch on j according to the following exhaustive subcases.
   (a) $c_i = c_l = c_m = 1$
   (b) $c_i = c_l = 1$, $c_m = 2$
   (c) $c_i = c_l = 1$, $c_m = 3$
   (d) $c_i = 1$, $c_l = c_m = 2$ with l, m adjacent
   (e) $c_i = 1$, $c_l = c_m = 2$ with l, m non adjacent
   (f) $c_i = 1$, $c_l = 2$, $c_m = 3$
   (g) $c_i = c_l = 2$, $c_m = 3$ with i, l adjacent
   (h) $c_i = c_l = 2$, $c_m = 3$ with i, l non adjacent
   (i) $c_i = 2$, $c_l = c_m = 3$
2. else $c_j \leqslant 2$ and `MAXKVC-3` is polynomially solvable.

Theorem 2. *Algorithm `MAXKVC-3` solves* MAX k-VERTEX COVER *on graphs with maximum degree* 3 *with running time* $O^*(1.3339^n)$.

Proof. Similarly to the general case, we count with a weight w_h the free vertices h, where each free vertex h is assigned a weight $w_h = w_{[i]}$ with $i = c_h = d_h - \alpha_h$. Also, we impose $w_{[0]} = 0$, $w_{[1]} = 0.2909$, $w_{[2]} = 0.5818$ and $w_{[3]} = 1$, hence $w_j = 1$. Notice that with these values we have $w_{[1]} = w_{[2]} - w_{[1]} = 0.2909 < w_{[3]} - w_{[2]} = 0.4182$. The overall complexity is then due to the worst-case among subcases 1a, 1b, 1c, 1d, 1e, 1f, 1g, 1h and 1i.

Subcase 1a. Vertices i, j, l, m form an independent subgraph disjoint from the rest of the graph. Then, as in the case of independent cycles and paths described in the last paragraph of Section 2.1, the optimal solution of this subgraph for any value of k can be computed with constant complexity and taken aside to be merged to the solution of the rest of graph. Then, with respect to the remaining graph, i, j, l, m are fixed without branching.

Subcase 1b. We branch on vertex m and either we select or discard m. In both cases, i, j, l form a disconnected subgraph and can be fixed similarly to subcase 1a, while the other neighbor q of m decreases its coefficient by one unit where the worst-case occurs when $c_q = 2$ as, for $c_q = 1$, vertices i, j, l, m, q would all together form an independent subgraph and no branching would occur. Hence, the recursion is $T(p) \leqslant 2T(p - w_{[3]} - w_{[2]} - 2w_{[1]} - (w_{[2]} - w_{[1]})) \approx 2T(p - 2.4545)$ with complexity $O^*(1.3263^n)$.

Subcase 1c. We branch on vertex m and either we select or discard m. In both cases, i, j, l form a disconnected subgraph and can be fixed similarly to subcase 1a, while the other neighbors q and r of m decrease their coefficients by one unit where the worst-case occurs when $c_q = 1$ and $c_r = 2$ as for $c_q = c_r = 1$ vertices i, j, l, m, q, r would all together form an independent subgraph and no branching would occur. Hence, the recursion is $T(p) \leqslant 2T(p - 2w_{[3]} - 3w_{[1]} - (w_{[2]} - w_{[1]})) \approx 2T(p - 3.1636)$ with complexity $O^*(1.245^n)$.

Subcase 1d. Vertices i, j, l, m form an independent subgraph disjoint from the rest of the graph and the same consideration of subcase 1a holds.

Subcase 1e. We branch on vertex j and either we select or discard j. Notice that, as $c_l = 2$, either $d_l = 3$ and $\alpha_l = 1$, or $d_l = 2$ and $\alpha_l = 0$. Now, for $d_l = 2$, if we select j, we have $d_l = 1$ and $\alpha_l = 0$ and, due to Lemma 1, i can be discarded and therefore fixed. Similarly, for $d_l = 3$, if we discard j, we have $d_l = 3$ and $\alpha_l = 2$ and, due to Lemma 2, l can be selected and therefore fixed. Overall, l will necessarily be fixed in one of the branches and correspondingly its neighbor decreases the coefficient by one unit. Similar consideration occurs for m as also $c_m = 2$. But then, the worst-case occurs when l and m are fixed in the same branch. Correspondingly, the recursion is $T(p) \leqslant T(p - w_{[3]} - 2w_{[2]} - 2w_{[1]} - w_{[1]}) + T(p - w_{[3]} - 2(w_{[2]} - w_{[1]}) - w_{[1]}) \approx T(p - 3.0363) + T(p - 1.8727)$ with complexity $O^*(1.3339^n)$.

Subcase 1f. We branch on vertex m and either we select or discard m. Notice that, as $c_i = 1$, we have $d_i = 2$ and $\alpha_i = 1$ or else i would be fixed without branching either due to Lemma 1 or to Lemma 2. But then, when m is selected, we have $d_j = 2$ with $alpha_j = 0$ with i and j adjacents: this implies that j is dominated by i and that i and j cannot be selected together, that is j can be discarded. Notice also, that when m is fixed the other two neighbors q and r decrease their coefficient by one unit where the worst case occurs for $c_q = c_r = 1$. Correspondingly, the recursion is $T(p) \leqslant T(p - 2w_{[3]} - (w_{[2]} - w_{[1]}) - 3w_{[1]}) + T(p - w_{[3]} - (w_{[3]} - w_{[2]}) - 2w_{[1]}) \approx T(p - 3.1636) + T(p - 2)$ with complexity $O^*(1.3144^n)$.

Subcase 1g. We branch on vertex m and either we select or discard m. In both cases, i, j, l form a disconnected subgraph and can be fixed, while the other neighbors q and r of m decrease their coefficients by one unit where the worst-case occurs when $c_q = 1$ and $c_r = 2$ as for $c_q = c_r = 1$ vertices i, j, l, m, q, r would all together form an independent subgraph and no branching would occur. Hence, the recursion is $T(p) \leqslant 2T(p - 2w_{[3]} - 2w_{[2]} - w_{[1]} - (w_{[2]} - w_{[1]})) \approx 2T(p - 3.7454)$ with complexity $O^*(1.2033^n)$.

Subcase 1h. We branch on vertex j and either we select or discard j. Notice that, as in subcase 1e, overall, l will necessarily be fixed in one of the branches and correspondingly its neighbor will decrease the coefficient by one unit. Similar consideration occurs for i as also $c_i = 2$. But then, the worst-case occurs when i and l are fixed in the same branch. Correspondingly, the recursion is $T(p) \leqslant T(p - w_{[3]} - 2w_{[2]} - 2w_{[1]} - (w_{[3]} - w_{[2]})) + T(p - w_{[3]} - 2(w_{[2]} - w_{[1]}) - (w_{[3]} - w_{[2]})) \approx T(p - 3.1636) + T(p - 2)$ with complexity $O^*(1.3144^n)$.

Subcase 1i. We branch on vertex j and either we select or discard j. Notice that, as in subcase 1e, overall, i will necessarily be fixed in one of the branches and correspondingly its neighbor will decrease the coefficient by one unit. Correspondingly, the recursion is $T(p) \leqslant T(p - w_{[3]} - w_{[2]} - w_{[1]} - 2(w_{[3]} - w_{[2]})) + T(p - w_{[3]} - 2(w_{[3]} - w_{[2]}) - (w_{[2]} - w_{[1]})) \approx T(p - 2.7091) + T(p - 2.1273)$ with complexity $O^*(1.3339^n)$.

The worst-case is then given by subcases 1e and 1i with complexity $O^*(1.3339^n)$. ∎By solving (8) for $\beta = 1.3339$ and $w_1 = 0.2909$, we get $\alpha = 0.081$ and, correspondingly, the following proposition holds and concludes the section.

Proposition 1. *By applying the memorization approach of section 2.1,* `MAXKVC-3` *solves* MAX k-VERTEX COVER *on graphs with maximum degree* 3 *with running time* $O^*(1.3249^n)$.

3 MAX k-VERTEX COVER and fixed parameter tractability

Denote by $(a - \overline{b} - c)$, a branch of the search tree where vertices a and c are selected and vertex b is discarded. Consider the vertex j with maximum degree Δ and neighbors $l_1, \ldots, l_\Delta$. As j has maximum degree, we may assume that if there exists an optimal solution of the problem where all neighbors of j are discarded, then there exists at least one optimal solution where j is selected. Hence, a branching scheme (called *basic branching scheme*) on j of type:

$$\left[l_1, \left(\overline{l_1} - l_2\right), \left(\overline{l_1} - \overline{l_2} - l_3\right), \ldots, \left(\overline{l_1} - \overline{l_2} - \ldots - \overline{l_{\Delta-1}} - l_\Delta\right), \left(\overline{l_1} - \overline{l_2} - \ldots - \overline{l_\Delta} - j\right)\right]$$

can be applied.

Proposition 2. *The* MAX k-VERTEX COVER *problem can be solved to optimality in* $O^*(\Delta^k)$. *In other words,* MAX k-VERTEX COVER *in bounded degree graphs is in* $\mathbf{FPT}(\mathbf{k})$.

Proof. Consider vertex j with maximum degree Δ and neighbors $l_1, \ldots, l_\Delta$ where the basic branching scheme of type $[l_1, (\overline{l_1} - l_2), (\overline{l_1} - \overline{l_2} - l_3), \ldots, (\overline{l_1} - \overline{l_2} - \ldots - \overline{l_{\Delta-1}} - l_\Delta), (\overline{l_1} - \overline{l_2} - \ldots - \overline{l_\Delta} - j)]$ can be applied. Then, the last two branches can be substituted by the branch $(\overline{l_1} - \overline{l_2} - \ldots - \overline{l_{\Delta-1}} - j)$ as, if all neighbors of j but one are not selected, any solution including the last neighbor l_Δ but not including j is not better than the solution that selects j.

Now, one can see that the basic branching scheme generates Δ nodes. On the other hand, we know that in each branch of the basic branching scheme at least one vertex is selected. As, at most k nodes can be selected, the overall complexity cannot be superior to $O^*(\Delta^k)$. ∎

Now, let $V' \subset V$ be a minimum vertex cover of G and let τ be the size of V' that is $\tau = |V'|$. Correspondingly, let $I = V \setminus V'$ be a maximum independent set of G and set $\alpha = |I|$. Notice that V' can be computed, for instance, in $O^*(1.28^\tau)$ time by means of the fixed parameter algorithm of [10]. Let us note that we can assume $k \leqslant \tau$. Otherwise, the optimal value ℓ for MAX k-VERTEX COVER would be equal to $|E|$ and one could compute a minimum vertex cover V' in G and then one could arbitrarily add $k - \tau$ vertices without changing the value of the optimal solution.

Theorem 3. MAX k-VERTEX COVER $\in \mathbf{FPT}(\tau)$.

Proof. In order to prove the claim, it suffices to prove that MAX k-VERTEX COVER can be solved to optimality in at most $O^*(2^\tau)$ time.

Fix some minimum vertex cover V' of G and consider some solution K for MAX k-VERTEX COVER, i.e., some set of k vertices of G. Any such set is distributed over V' and its associated independent set $I = V \setminus V'$. Fix now an optimal solution K^* of MAX k-VERTEX COVER and denote by S' the subset of V' that belongs to K^* (S' can be eventually the empty set) and by I' the part of K^* belonging to I. In other words, the following hold:

$$\begin{array}{rcl} K^* & = & S' \cup I' \\ S' & \subseteq & V' \\ I' & \subseteq & I \;=\; V \setminus V' \end{array}$$

Given S' (assume $|S'| = k'$), it can be completed into K^* in polynomial time. Indeed, for each vertex i belonging to I we need simply to compute (in linear time) the total number e_i of edges (i, j) for all $j \in V' \setminus S'$. Then, I' is obtained by selecting the $k - k'$ vertices of I with largest e_i value. So, the following algorithm can be used for MAX k-VERTEX COVER:

1. compute a minimum vertex cover V' (using the algorithm of [10]);
2. for every subset $S' \subseteq V'$ of cardinality at most k, take the $k - |S'|$ vertices of $V \setminus V'$ with the largest degrees to $V' \setminus S'$; denote by I' this latter set;
3. return the best among the sets $S' \cup I'$ so-computed (i.e., the set that covers the maximum of edges).

Step 1 takes time $O^*(1.28^\tau)$, while step 2 has total running time $O^*(\sum_{i=1}^{k}\binom{\tau}{i}) < O^*(2^\tau)$. ∎

Note that, from Theorem 3 it can be immediately derived that MAX k-VERTEX COVER can be solved to optimality in $O^*(2^{\frac{\Delta-1}{\Delta}n})$ time. Indeed if a graph G has maximum degree Δ, then for the maximum independent set we have $\alpha \geqslant \frac{n}{\Delta}$. Also, we can assume that G is not a clique on $\Delta+1$ vertices (note that MAX k-VERTEX COVER is polynomial in cliques). In this case, G can be colored with Δ colors [7]. In such a coloring the cardinality of the largest color is greater than $\frac{n}{\Delta}$ and, a fortiori, so is the cardinality of a maximum independent set (since each color is an independent set). Consequently, $\tau \leqslant \frac{\Delta-1}{\Delta}n$.

We now try to have some further insight into the inclusion of MAX k-VERTEX COVER in several fixed parameter tractability classes, mainly the class **FPT(k)**, as well as, into the possibility to improve Theorem 3. In what follows we show a rather interesting result claiming, informally, that the instances of MAX k-VERTEX COVER that are not fixed parameter tractable (with respect to k) are those solved with running time better than $O^*(2^\tau)$.

Theorem 4. *For every instance (G,k) of* MAX k-VERTEX COVER*, either $(G,k) \in \mathbf{FPT}(\mathbf{k})$, or it can be solved in time $O^*(\gamma^\tau)$, for some $\gamma < 2$.*

Proof. Recall that the running time of the algorithm in the proof of Theorem 3 is $O^*(\sum_{i=1}^{k}\binom{\tau}{i})$. As mentioned above, k can be assumed to be smaller than, or equal to, τ. Consider some positive constant $\lambda < 1/2$. We distinguish the following two cases: $\tau > k \geqslant \lambda\tau$ and $k < \lambda\tau$.

If $\tau > k \geqslant \lambda\tau$, then $\tau \leqslant k/\lambda$. As $\lambda < 1/2$, $k/\lambda > 2k$ and, since $i \leqslant k$, we get using Stirling's formula:

$$\sum_{i=1}^{k}\binom{\tau}{i} \leqslant \sum_{i=1}^{k}\binom{k/\lambda}{i} \leqslant k\binom{k/\lambda}{k} \sim k\frac{\frac{k}{\lambda}^{\frac{k}{\lambda}}}{k^k\left(\frac{k}{\lambda}-k\right)^{\left(\frac{k}{\lambda}-k\right)}} = k\left(\frac{\frac{1}{\lambda}^{\frac{1}{\lambda}}}{\left(\frac{1}{\lambda}-1\right)^{\left(\frac{1}{\lambda}-1\right)}}\right)^k = O^*\left(c^k\right) \qquad (9)$$

for some constant c that depends on λ and it is fixed if λ is so.

If $k < \lambda\tau$, then, by the hypothesis on λ, $2k < \tau$ and, since $i \leqslant k$, expression $\sum_{i=1}^{k}\binom{\tau}{i}$ is bounded above by $k\binom{\tau}{k}$. In all, using also Stirling's formula the following holds:

$$\sum_{i=1}^{k}\binom{\tau}{i} \leqslant k\binom{\tau}{k} \leqslant k\binom{\tau}{\lambda\tau} \sim k\frac{\tau^\tau}{(\lambda\tau)^{(\lambda\tau)}[(1-\lambda)\tau]^{(1-\lambda)\tau}} = k\left(\frac{1}{\lambda^\lambda(1-\lambda)^{(1-\lambda)}}\right)^\tau \overset{\lambda<1/2}{<} O^*\left(2^\tau\right) \qquad (10)$$

In other words, if $k < \lambda\tau$, then MAX k-VERTEX COVER can be solved in time $O^*(\gamma^\tau)$, for some γ that depends on λ and is always smaller than 2 for $\lambda < 1/2$.

λ	0.01	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.40	0.45	0.49
$\frac{\frac{1}{\lambda}^{\frac{1}{\lambda}}}{\left(\frac{1}{\lambda}-1\right)^{\left(\frac{1}{\lambda}-1\right)}}$	270.47^k	53.00^k	25.81^k	16.74^k	12.21^k	9.48^k	7.66^k	6.36^k	5.38^k	4.61^k	4.11^k
$\frac{1}{\lambda^\lambda(1-\lambda)^{1-\lambda}}$	1.06^τ	1.22^τ	1.38^τ	1.53^τ	1.65^τ	1.75^τ	1.84^τ	1.91^τ	1.96^τ	1.99^τ	1.9996^τ

Table 3: The values of c and γ for some values of λ.

Expressions (9) and (10) derive the claim and conclude the proof. In Table 3 the values of c and γ are given for some values of λ. ∎

Observe now that $k \leqslant \ell$ (the optimal value for MAX k-VERTEX COVER). Indeed, in any non-trivial solution K, any of its vertices covers at least one edge that is not covered by any other vertex of K. Hence, the following corollary holds for the standard fixed parameter tractability (i.e., with respect to ℓ) of MAX k-VERTEX COVER.

Corollary 1. *For every instance (G,k) of* MAX k-VERTEX COVER*, either $(G,k) \in \mathbf{FPT}(\ell)$, or it can be solved in time $O^*(\gamma^\tau)$, for some $\gamma < 2$.*

Let us note that the technique of Theorem 3, that consists of taking a subset S' of a minimum vertex cover V' of the input graph and of completing it into an optimal solution can be applied in several other well-known combinatorial **NP**-hard problems. We sketch here some examples:

- in MIN 3-DOMINATING SET (dominating set in graphs of maximum degree 3, S' is completed in the following way:
  - take all the vertices in $I \setminus \Gamma_I(S')$ (in order to dominate vertices in $V' \setminus S'$;
  - if there remain vertices of $V' \setminus S'$ not dominated yet solve a MIN SET COVER problem considering $\Gamma_I(S')$ as the set-system of the latter problem and assuming that a vertex $v \in \Gamma_I(S')$, seen as set, contains its neighbors in $V' \setminus S'$ as elements; since $\Gamma_I(S')$ is the neighborhood of S', the degrees of its vertices to $V' \setminus S'$ are bounded by 2, that induces a polynomial MIN SET COVER problem ([21]);
- in MIN INDEPENDENT DOMINATING SET, S' is completed by the set $I \setminus \Gamma_I(S')$, where $\Gamma_I(S')$ is the set of neighbors of S' that belong to I;
- in EXISTING DOMINATING CLIQUE, MIN DOMINATING CLIQUE (if any), MAX DOMINATING CLIQUE (if any) and MAX CLIQUE, S' can eventually be completed by a single vertex of $\Gamma_I(S')$.

Theorem 5. MIN INDEPENDENT DOMINATING SET, EXISTING DOMINATING CLIQUE, MIN DOMINATING CLIQUE, MAX DOMINATING CLIQUE, MAX CLIQUE *and* MIN 3-DOMINATING SET *belong to* $\mathbf{FPT}(\tau)$.

4 MAX k-VERTEX COVER in bipartite graphs

MAX k-VERTEX COVER belongs to a large class of optimization problems called "cardinality constrained optimization problems" in [8]. Two other notable representatives of this class are the DENSEST k-SUBGRAPH and the SPARSEST k-SUBGRAPH problems. Furthermore, the following equivalence is proved in [8].

Lemma 3. ***([8])*** *Given a graph G,* DENSEST k-SUBGRAPH *in G,* SPARSEST k-SUBGRAPH *in $\bar{G}$ and* MAX $(n-k)$-VERTEX COVER *in $\bar{G}$ are equivalent.*

On the other hand, the DENSEST k-SUBGRAPH problem is known to be **NP**-hard in bipartite graphs [11]. The reduction there implies also that the DENSEST k-SUBGRAPH PROBLEM is **NP**-hard even if we seek for a solution consisting of q vertices from the first color class of the bipartite graph and $\binom{q}{2}$ vertices from the second one, such that $k = q + \binom{q}{2}$. We call this problem DENSEST $(q, \binom{q}{2})$-SUBGRAPH problem.

Lemma 4. *The* DENSEST k-SUBGRAPH *problem is **NP**-hard in complements of bipartite graphs.*

Proof. Assume that there exists a polynomial algorithm that finds an optimal solution for the DENSEST $(q, \binom{q}{2})$-SUBGRAPH problem in complements of bipartite graphs. Then, such an optimum leads to an optimal solution for the densest $(q, \binom{q}{2})$-subgraph problem in the bipartite graph that results from removing the edges inside the two cliques of the initial graph. To see this, note that the number of edges of the optimal solution inside the two cliques is fixed and equal to $\binom{q}{2} + \binom{\binom{q}{2}}{2}$. In other words, the value of the optimal depends only on the edges between the two cliques, that is between the two classes of the bipartite graph. Hence the DENSEST k-SUBGRAPH problem is **NP**-hard for complements of bipartite graphs. ∎

Putting together Lemmata 3 and 4, the following result holds and concludes the section.

Proposition 3. *The* MAX k-VERTEX COVER *problem is **NP**-hard in bipartite graphs.*

Note that Theorem 3 in Section 3, derives immediately the following result for MAX k-VERTEX COVER in bipartite graphs (recall that in these graphs $\tau \leqslant n/2$).

Corollary 2. MAX k-VERTEX COVER *can be solved in $O^*(2^{n/2}) = O^*(1.414^n)$ in bipartite graphs.*

5 Approximating MAX k-VERTEX COVER by moderately exponential algorithms

We now show how one can get approximation ratios non-achievable in polynomial time using moderately exponential algorithms with worst-case running times better than those required for an exact computation (see [3, 4] for more about this issue). Denote by $\mathrm{opt}(G)$ the cardinality of an optimal solution for MAX

k-VERTEX COVER in G and by $m(G)$, the cardinality of an approximate solution. Our goal is to study the approximation ratio $m(G)/opt(G)$.

In what follows, we denote, as previously, by K^* the optimal solution for MAX k-VERTEX COVER. Given a set K of vertices, we denote by $C(K)$, the set of edges covered by K (in other words, the value of a solution K for MAX k-VERTEX COVER is $|C(K)|$; also, according to our previous notation, $\mathrm{opt}(G) = |C(K^*)|$). We first prove the following easy lemma that will be used later.

Lemma 5. *For any $\lambda \in [0,1]$, the subset H^* of λk vertices of K^* covering the largest amount of edges covered by K^*, covers more than $\lambda\,\mathrm{opt}(G)$ edges.*

Proof. Indeed, if the λk "best" vertices of K^* cover less than $\lambda\,\mathrm{opt}(G)$ edges, then any disjoint union of k/λ subsets of K^*, each of cardinality λk covers less than $\mathrm{opt}(G)$ edges, a contradiction. ∎

Now, for some $\lambda \in [0,1]$, run the following algorithm, denoted by `APMAXKVC1`:

1. take all the combinations $C_n^{\lambda k}$ of λk vertices among n;
2. for any of the sets K' produced in step 1, remove K' and $C(K')$ from G and run a ρ-approximation algorithm in the surviving graph in order to compute a solution K'' for the MAX $(1-\lambda)k$-VERTEX COVER problem;
3. output the best $\hat{K} = \hat{K}' \cup \hat{K}''$, i.e., the one that covers the most of edges, produced in step 2.

It is easy to see that the complexity of `APMAXKVC1` is bounded above by $O^*(n^{\lambda k})$. Furthermore, it can be implemented to run with polynomial space by simply storing the best current K.

Proposition 4. MAX k-VERTEX COVER *can be solved within approximation ratio $1-\epsilon$, for any $\epsilon > 0$, in time $O^*(n^{(1-4\epsilon)k})$ and with polynomial space.*

Proof. Fix an optimal solution K^* for MAX k-VERTEX COVER ($|K^*| = k$) in G. Denote by K_1^* the λk vertices of K^* that cover the most of edges of $C(K^*)$. Since K_1^* has been produced by algorithm `APMAXKVC1` in step 1, if $C(K_2^*)$ is the value of the solution for MAX $(1-\lambda)k$-VERTEX COVER, associated with K_1* produced in step 2, the following holds:

$$\left|C\left(\hat{K}' \cup \hat{K}''\right)\right| = \left|C\left(\hat{K}'\right)\right| + \left|C\left(\hat{K}''\right)\right| \geqslant |C\left(K_1^* \cup K_2^*\right)| = |C\left(K_1^*\right)| + |C\left(K_2^*\right)| \tag{11}$$

Consider the subgraph $G_1 = G[V \setminus K_1^*]$ of G induced by $V \setminus K_1^*$. There, the set $K'^* = K^* \setminus K_1^*$ is a feasible solution for MAX $(1-\lambda)k$-VERTEX COVER. Denoting by $\mathrm{opt}_{(1-\lambda)k}(G_1)$ the value of an optimal solution for MAX $(1-\lambda)k$-VERTEX COVER in G_1, we have:

$$|C\left(K'^*\right)| = \mathrm{opt}(G) - |C\left(K_1^*\right)| \leqslant \mathrm{opt}_{(1-\lambda)k}\left(G_1\right) \tag{12}$$

where, as previously, $\mathrm{opt}(G) = |C(K^*)|$, is the optimal value for MAX k-VERTEX COVER in G.

Recall also that, by step 2, $C(K_2^*)$ satisfies:

$$|C\left(K_2^*\right)| \geqslant \rho \underset{(1-\lambda)k}{\mathrm{opt}}\left(G_1\right) \tag{13}$$

Putting together (12) and (13) we get:

$$\begin{aligned} \frac{|C\left(K_1^*\right)| + |C\left(K_2^*\right)|}{\mathrm{opt}(G)} &\geqslant \frac{|C\left(K_1^*\right)| + \rho\,\mathrm{opt}_{(1-\lambda)k}\left(G_1\right)}{\mathrm{opt}(G)} \geqslant \frac{|C\left(K_1^*\right)| + \rho\left(\mathrm{opt}(G) - |C\left(K_1^*\right)|\right)}{\mathrm{opt}(G)} \\ &= \frac{\rho\,\mathrm{opt}(G) + (1-\rho)\,|C\left(K_1^*\right)|}{\mathrm{opt}(G)} \end{aligned} \tag{14}$$

By Lemma 5, $|C(K_1^*)| \geqslant \lambda\,\mathrm{opt}(G)$ and combining it with (11) and (14), we get:

$$\frac{m(G)}{\mathrm{opt}(G)} \geqslant \frac{\left|C\left(\hat{K}' \cup \hat{K}''\right)\right|}{\mathrm{opt}(G)} \geqslant \frac{|C\left(K_1^*\right)| + |C\left(K_2^*\right)|}{\mathrm{opt}(G)} \geqslant \rho + \lambda(1-\rho) \tag{15}$$

Taking $\rho = \frac{3}{4}$, the best known polynomially achieved ratio for MAX k-VERTEX COVER ([23]), in order to get an approximation ratio $1-\epsilon$, for some $\epsilon > 0$, one has to choose $\lambda = 1 - 4\epsilon$ and the proposition follows. ∎

We now improve the result of Proposition 4 by the following algorithm, called `APMAXKVC2` in what follows:

1. fix some $\lambda \in [0,1]$ and optimally solve MAX λk-VERTEX COVER in G (as previously, let H^* be the optimal solution built and $C(H^*)$ be the edge-set covered by H^*);
2. remove H^* and $C(H^*)$ from G and approximately solve MAX $(1-\lambda)k$-VERTEX COVER in the surviving graph (by some approximation algorithm); let K' be the obtained solution;
3. output $K = H^* \cup K'$.

It is easy to see that if $T(p,k)$ is the running time of an optimal algorithm for MAX k-VERTEX COVER, where p is some parameter of the input-graph G (for instance, n, or τ), then the complexity of `APMAXKVC2` is $T(p,\lambda k)$. Furthermore, `APMAXKVC2` requires polynomial space.

Theorem 6. *If $T(p,k)$ is the running time of an optimal algorithm for* MAX k-VERTEX COVER*, then, for any $\epsilon > 0$,* MAX k-VERTEX COVER *can be approximated within ratio $1-\epsilon$ and with worst-case running time $T(p,(1+2\sqrt{1-3\epsilon})k/3)$.*

Proof. Denote by K^* an optimal solution of MAX k-VERTEX COVER in G, by G_2 the induced subgraph $G[V \setminus H^*]$ of G, by $\mathrm{opt}_{(1-\lambda)}(G_2)$, the value of an optimal for MAX $(1-\lambda)k$-VERTEX COVER in G_2. Suppose that E' edges are common between $C(H^*)$ and $C(K^*)$. This means that $C(K^*) \setminus E'$ edges of $C(K^*)$ are in G_2 and are exclusively covered by the vertex-set $L^* = K^* \setminus H^*$ that belongs to G_2. Set $\ell^* = |L^*|$ and note that $\ell^* \leqslant k$ and $\ell^* \geqslant (1-\lambda)k$.

According to Lemma 5, the $(1-\lambda)k$ "best" vertices of L^* cover more than $(1-\lambda)|C(K^*) \setminus E'| = (1-\lambda)(\mathrm{opt}(G)-|E'|)$ edges in G_2 and these vertices constitute a feasible solution for MAX $(1-\lambda)k$-VERTEX COVER in G_2. Hence:

$$\mathrm{opt}_{(1-\lambda)}(G_2) \geqslant (1-\lambda)(\mathrm{opt}(G) - |E'|) \tag{16}$$

Taking into account (16), the fact that K' in step 2 of `APMAXKVC2` has been computed by, say, a ρ-approximation algorithm and the fact that $|E'| \leqslant |C(H^*)|$, we get:

$$\begin{aligned} m(G) &= C(H^*) + C(K') \geqslant C(H^*) + \rho(1-\lambda)\mathrm{opt}_{(1-\lambda)}(G_2) \\ &\geqslant C(H^*) + \rho(1-\lambda)(\mathrm{opt}(G) - |E'|)\, C(H^*) + \rho(1-\lambda)(\mathrm{opt}(G) - C(H^*)) \\ &\geqslant (1-\rho(1-\lambda))C(H^*) + \rho(1-\lambda)\,\mathrm{opt}(G) \end{aligned} \tag{17}$$

Using once more Lemma 5, $|C(H^*)| \geqslant \lambda\,\mathrm{opt}(G)$, and putting it together with (17), we get:

$$\frac{m(G)}{\mathrm{opt}(G)} \geqslant \rho(1-\lambda) + \lambda(1-\rho(1-\lambda)) \tag{18}$$

Setting $\rho = \frac{3}{4}$ in (18), in order to achieve an approximation ratio $m(G)/\mathrm{opt}(G) = 1-\epsilon$, for some $\epsilon > 0$, we have to choose an λ satisfying $\lambda = (1+2\sqrt{1-3\epsilon})/3$, that completes the proof of the theorem. ∎

Note that Proposition 4 and Theorem 6 reach the same complexity $O(n^{0.737k})$ for $\epsilon = 0.0657$. In Table 4, a comparative study of the running times claimed by Proposition 4 and Theorem 6 with respect to some ratio values is given.

ϵ	0.2	0.15	0.1	0.08	0.0657	0.05	0.02	0.01	0.001
Proposition 4	$n^{0.2k}$	$n^{0.4k}$	$n^{0.6k}$	$n^{0.68k}$	$n^{0.737k}$	$n^{0.8k}$	$n^{0.92k}$	$n^{0.96k}$	$n^{0.996k}$
Theorem 6	$n^{0.598k}$	$n^{0.656k}$	$n^{0.706k}$	$n^{0.724k}$	$n^{0.737k}$	$n^{0.751k}$	$n^{0.776k}$	$n^{0.784k}$	$n^{0.791k}$

Table 4: Complexity results for small values of ϵ provided by Proposition 4 and Theorem 6.

It can be seen from Table 4 that Proposition 4 dominates Theorem 6 for large ϵ's and is dominated by Theorem 6 for small ϵ's (the break being on $\epsilon = 0.0657$ inducing complexity $O^*(n^{0.737k})$. Notice, also, that Proposition 4 makes sense for $\epsilon \geqslant 0.05$ only, as for lower values of ϵ the computational cost claimed by Proposition 4, becomes superior to that of the exact approach proposed in [8].

Corollary 3. MAX k-VERTEX COVER *can be approximated within ratio $1-\epsilon$ and with running time:*

$$\min\left\{O^*\left(n^{\left(1+2\sqrt{1-3\epsilon}\right)(\omega k)/9}\right), O^*\left(n^{\frac{\tau}{\left(1+2\sqrt{1-3\epsilon}\right)k/3}}\right), O^*(n^{(1-4\epsilon)k})\right\}$$

For Corollary 3, just observe that the running-times claimed for the first two entries are those needed to optimally solve MAX λk-VERTEX COVER (the former due to [8] and the latter due to Theorem 3), while the last running time corresponds to the complexity of the approach proposed in Proposition 4.

Finally, let us close this section and the paper by some remarks on what kind of results can be expected in the area of (sub)exponential approximation. All the algorithms given in this section have exponential running time when we seek for a *constant* approximation ratio (unachievable in polynomial time). On the other hand, for several problems that are hard to approximate in polynomial time (like MAX INDEPENDENT SET, MIN COLORING, ...), subexponential time can be easily reached for ratios depending on the input size (thus tending to ∞, for minimization problems, or to 0, for maximization problems). An interesting question is to determine, for these problems, if it is possible to devise a constant approximation algorithm working in subexponential time. An easy argument shows that this is not always the case. For instance, the existence of subexponential approximation algorithms (within ratio better than 4/3) is quite improbable for MIN COLORING since it would imply that 3-COLORING can be solved in subexponential time, contradicting so the "exponential time hypothesis" [22]. We conjecture that this is true for any constant ratio for MIN COLORING. Anyway, the possibility of devising subexponential approximation algorithms for **NP**-hard problems, achieving ratios forbidden in polynomial time or of showing impossibility of such algorithms is an interesting open question that deserves further investigation.

References

[1] Lingo 2.0 - optimization modeling software for linear, nonlinear, and integer programming. http://www.lindo.com/index.php, 2010.

[2] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.* To appear in the special issue dedicated to selected papers from FOCS'06.

[3] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation by "low-complexity" exponential algorithms. Cahier du LAMSADE 271, LAMSADE, Université Paris-Dauphine, December 2007. Available at `http://www.lamsade.dauphine.fr/cahiers/PDF/cahierLamsade271.pdf`.

[4] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation of combinatorial problems by moderately exponential algorithms. In F. Dehne, M. Gavrilova, J.-R. Sack, and C. D. Tóth, editors, *Proc. Algorithms and Data Structures Symposium, WADS'09*, volume 5664 of *Lecture Notes in Computer Science*, pages 507–518. Springer-Verlag, 2009.

[5] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation of MIN COLORING by moderately exponential algorithms. *Inform. Process. Lett.*, 109(16):950–954, 2009.

[6] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation of MIN SET COVER by moderately exponential algorithms. *Theoret. Comput. Sci.*, 410(21-23):2184–2195, 2009.

[7] R. L. Brooks. On coloring the nodes of a network. *Math. Proc. Cambridge Philos. Soc.*, 37:194–197, 1941.

[8] L. Cai. Parameter complexity of cardinality constrained optimization problems. *The Computer Journal*, 51:102–121, 2008.

[9] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108. Springer-Verlag, 2006.

[10] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *J. Algorithms*, 41:280–301, 2001.

[11] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Appl. Math.*, 9:27–39, 1984.

[12] M. Cygan, L. Kowalik, and M. Wykurz. Exponential-time approximation of weighted set cover. *Inform. Process. Lett.*, 109(16):957–961, 2009.

[13] M. Cygan and M. Pilipczuk. Exact and approximate bandwidth. *Theoret. Comput. Sci.*, 411(40–42):3701–3713, 2010.

[14] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer, New York, 1999.

[15] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer-Verlag, 2006.

[16] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. 41:174–211, 2001.

[17] M. Fellows. Private communication, April 2011.

[18] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. Bulletin of the European Association for Theoretical Computer Science 87, pp. 47–77, 2005.

[19] F. V. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. Assoc. Comput. Mach.*, 56(5):1–32, 2009.

[20] M. Fürer, S. Gaspers, and S. P. Kasiviswanathan. An exponential time 2-approximation algorithm for bandwidth. In *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'09*, volume 5917 of *Lecture Notes in Computer Science*, pages 173–184. Springer, 2009.

[21] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[22] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001.

[23] G. Jager and A. Srivastav. Improved approximation algorithms for maximum graph partitioning problems. In *Proc. Foundations of Software Technology and Theoretical Computer Science, FST&TCS'04*, volume 3328 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, 2004.

[24] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Comment. Math. Univ. Carolinae*, pages 415–419, 1985.

[25] R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. 47(2):63–77, 2003.