Approximation of Knapsack Problems with Conflict and Forcing Graphs[☆]

Ulrich Pferschy[a], Joachim Schauer[a]

[a] *University of Graz, Institute of Statistics and Operations Research, Universitaetsstr. 15, 8010 Graz, Austria,* `{pferschy, joachim.schauer}@uni-graz.at`

Abstract

We study the classical 0-1 knapsack problem with additional restrictions on pairs of items. A conflict constraint states that from a certain pair of items at most one item can be contained in a feasible solution. Reversing this condition, we obtain a forcing constraint stating that at least one of the two items must be included in the knapsack. A natural way for representing these constraints is the use of conflict (resp. forcing) graphs.

By modifying a recent result of Lokstanov et al. (2014) we derive a fairly complicated FPTAS for the knapsack problem on weakly chordal conflict graphs. Next, we show that the techniques of modular decompositions and clique separators, widely used in the literature for solving the independent set problem on special graph classes, can be applied to the knapsack problem with conflict graphs. In particular, we can show that every positive approximation result for the atoms of prime graphs arising from such a decomposition carries over to the original graph. We point out a number of structural results from the literature which can be used to show the existence of an FPTAS for several graph classes characterized by the exclusion of certain induced subgraphs. Finally, a PTAS for the knapsack problem with H-minor free conflict graph is derived. This includes planar graphs and, more general, graphs of bounded genus. The PTAS is obtained by expanding a general result of Demaine et al. (2005).

The knapsack problem with forcing graphs can be transformed into a minimization knapsack problem with conflict graphs. It follows immediately that all our FPTAS results of the current and a previous paper carry over from conflict graphs to forcing graphs. In contrast, the forcing graph variant is already inapproximable on planar graphs.

Keywords: knapsack problem, conflict graph, weakly chordal graph, planar graph, graph decomposition
2010 MSC: 90C27, 05C85

[☆]This research was funded by the Austrian Science Fund (FWF): P23829

1. Introduction

The classical 0-1 knapsack problem is an $\mathcal{NP}$-hard discrete optimization problem known to be relatively easy to solve in practice. This pleasant behavior usually changes as soon as additional constraints are imposed in addition to the standard weight constraint. In this paper we consider disjunctive constraints on pairs of items as a structurally simple but highly relevant class of conditions. In particular, we will study the following two types of restrictions:

- A *conflict constraint* (negative disjunctive constraint) on a pair of items expresses an incompatibility between these items. For each conflicting pair, *at most* one item can occur in a feasible knapsack solution.
- A *forcing constraint* (positive disjunctive constraint) enforces that *at least* one item from the underlying pair of items has to be included in a feasible solution.

It is natural to represent these conflict and forcing constraints by means of an undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, where each vertex corresponds uniquely to one item and an edge $(i, j) \in E$ indicates that items i and j are in a conflict (resp. forcing) relation.

Introducing the standard knapsack problem KP with n items, each of them with profit p_j and weight w_j, $j = 1, \ldots, n$, and a knapsack capacity c (cf. Kellerer et al. [24]), we obtain the *knapsack problem with conflict graph* (KCG) by imposing the above conflict constraints on the solution of KP.

$$
\begin{aligned}
(KCG) \qquad & \max && \sum_{j=1}^{n} p_j x_j && (1)\\
& \text{s.t.} && \sum_{j=1}^{n} w_j x_j \le c && (2)\\
& && x_i + x_j \le 1 \quad \forall\ (i, j) \in E && (3)\\
& && x_j \in \{0, 1\} \quad j = 1, \ldots, n. && (4)
\end{aligned}
$$

It is sometimes also referred to as *disjunctively constrained knapsack problem*, e.g. in the first paper dealing with KCG due to Yamada et al. [39] where a branch-and-bound algorithm was presented. For more recent work on exact algorithms and heuristics for KCG see Hifi and Otmani [22] and Hifi et al. [23] and the references therein.

Conflict graphs were also considered for many other combinatorial optimization problems such as bin packing (see e.g. Muritiba et al. [30] and Sadykov and Vanderbeck [35]), scheduling problems (see e.g. Even et al. [16]). Recently, positive and negative disjunctive constraints were considered for MST, matching and shortest path in Darmann et al. [13], for matching in Öncan et al. [31], for

MST in Zhang et al. [40] and for the maximum flow problem in Pferschy and Schauer [33].

A related problem is the *knapsack problem with forcing graph* (KFG) which was, to the best of our knowledge, not considered in the literature before. Given a forcing graph $G = (V, E)$ it follows from KCG by replacing (3) by

$$x_i + x_j \geq 1 \quad \forall \ (i, j) \in E. \tag{5}$$

We can also take a different view at these two problems from a graph theoretic perspective. Clearly, KCG is equivalent to a *maximum weight independent set problem* ($MWIS$) maximizing the sum of p_j with an additional weight or budget constraint given by the w_j. $MWIS$ is known as a famous $\mathcal{NP}$-hard problem and even hard to approximate within a factor of $n^{1-\varepsilon}$ even in the unweighted case with $p_j = 1$ (Håstad [19]).

A large body of literature exists for identifying special graph classes where $MWIS$ is still polynomially solvable or allows a positive approximation result. In this paper we will proceed in the same direction for KCG, i.e. for $MWIS$ with a budget constraint.

Problem KFG is closely related to a *minimum weight vertex cover problem* but with a different objective function. We will come back to this relation in Section 5. For minimum weight vertex cover problem it is known that no PTAS exists, but various papers study the approximability on special graph classes. Both independent set and vertex cover are polynomially solvable on perfect graphs. In contrast, it can be shown that both KCG and KFG are strongly $\mathcal{NP}$-hard for perfect graphs.

Because of this inherent difficulty of KCG and KFG we concentrate in this paper on the identification of special graph classes as conflict (resp. forcing) graphs to derive positive approximation results. In Pferschy and Schauer [32] we derived FPTASs for KCG on graphs of bounded treewidth and on chordal graphs. These are both based on tree structures representing the graph G and dynamic programming. In this paper we make considerable progress by applying much more complicated methods to derive approximation results for larger graph classes.

The contributions of this paper are the following:

1. An important generalization of chordal graphs are weakly chordal graphs. It will be shown in Section 2 that KCG still permits an FPTAS on weakly chordal graphs (and more generally on all graphs whose relevant potential maximal cliques can be listed in polynomial time, e.g. P_5-free graphs). The underlying technique of this construction was developed by Fomin and Villanger [17]. It was expanded by Lokshtanov et al. [25] who recently gave a polynomial time algorithm for the weighted independent set problem on P_5-free graphs. Note that this technique is crucial for KCG on weakly chordal graphs since it allows dynamic programming.

2. A general algorithmic strategy for solving difficult optimization problems on graphs is graph decomposition. A frequently applied pattern (see e.g. the extensive work by Brandstädt) applies modular decomposition followed by clique decomposition to break down the graph into much smaller and sometimes "benevolent" subgraphs. In fact, there are a number of graph classes for which special structural properties of the resulting *atoms of prime graphs* are known. We will show in Section 3 that KCG allows an FPTAS on any graph for which the atoms of its prime components allow an FTPAS. A number of examples for such graph classes is described in Section 3.3.
3. It is well known that planar graphs allow positive approximation results for vertex cover and independent set. Therefore, planar graphs are obvious candidates for seeking further positive results for disjunctively constrained knapsack problems. Indeed, we show in Section 4 that KCG allows a PTAS (but not an FPTAS) for H-minor free conflict graphs. We will more generally show that a result of Demaine et al. [14] can be expanded to guarantee PTASs for a more general class of graph problems. This result includes KCG on graphs of bounded genus.
4. Considering FCG in Section 5 it turns out that all results for an FPTAS carry over from KCG to KFG by means of a simple transformation. Interestingly, this analogy breaks down for planar graphs: In fact, even finding a feasible solution for KFG on planar graphs is strongly $\mathcal{NP}$-complete while there exists a PTAS for KCG.

Table 1 gives an overview of our main results for KCG and KFG.

graph class	KCG	KFG
general	strongly $\mathcal{NP}$-hard	feasibility $\mathcal{NP}$-complete
bounded treewidth	FPTAS	FPTAS
chordal	FPTAS	FPTAS
weakly chordal	FPTAS	FPTAS
planar	PTAS, no FPTAS	feasibility $\mathcal{NP}$-complete
perfect	strongly $\mathcal{NP}$-hard	strongly $\mathcal{NP}$-hard

Table 1: Complexity results for KCG and KFG on special graph classes.

1.1. Basic Definitions

In this section we give some definitions and concepts that are used throughout the paper. Mostly, we stick to standard notation as it is used in textbooks such as Diestel [15].

Definition. *Upper bound*

Throughout the paper we will frequently require an upper bound on the objective function value (1). While for practical applications the quality of an upper bound is crucial, a trivial bound of $P := \sum_{j=1}^{n} p_j$ *suffices for our theoretical results.*

Definition. *Neighborhood*

For a given subset of vertices A, the neighborhood $N[A]$ contains A and all vertices adjacent to at least one vertex in A (analogous for a single vertex v we have $N[v]$) and $N(A) := N[A] \setminus A$.

Definition. *ab-separator*

A set of vertices $V' \subset V$ is called ab-separator if the vertices a and b belonging to the same connected component of G are in different components of $G \setminus V'$.

Definition. *Separator*

A set $V' \subset V$ is a separator if there are vertices a and b in G such that V' is an ab-separator. A separator V' is a minimal ab-separator if any proper subset of V' is not an ab-separator in G. A separator V' is a minimal separator if there are vertices a and b in G such that V' is a minimal ab-separator.

Definition. *Tree-decomposition (cf. [15, Sec. 12])*

Let $G = (V, E)$ be a graph, T a tree, and let $\mathcal{V} = (V_I)_{I \in V(T)}$ be a family of vertex sets $V_I \subseteq V(G)$ indexed by the vertices I of T. By capital letters we refer to vertices from T, whereas by lower case letters we refer to vertices from G. The pair $(T, \mathcal{V})$ is called a tree-decomposition if it satisfies the following three properties:

1. *$V(G) = \bigcup_{I \in T} V_I$;*
2. *for every edge $e \in G$ there exists $I \in T$ such that both ends of e lie in V_I;*
3. *$V_{I_1} \cap V_{I_3} \subseteq V_{I_2}$ whenever I_2 lies on the path from I_1 to I_3 in T.*

Definition. *Treewidth*

The width of $(T, \mathcal{V})$ is defined as $\max\{|V_I| - 1 \mid I \in T\}$. The treewidth of G is the smallest width of any tree-decomposition of G.

Note that deciding whether a tree-decomposition of treewidth at most k exists, and if so, finding such a tree-decomposition (for constant k) can be done in linear time (cf. Bodlaender and Koster [3]).

Definition. *Graphs having a property* nearly *(cf. Brandstädt and Hoàng [8])*

For a given graph property $\mathcal{P}$ a graph is nearly *$\mathcal{P}$, if for all vertices v of G the graph induced by V without the neigborhood of v, i.e. $G[V \setminus N(v)]$, fulfills $\mathcal{P}$.*

For KCG it is easy to see that any pseudopolynomial algorithm for graphs with property $\mathcal{P}$ will trivially imply a pseudopolynomial algorithm for nearly $\mathcal{P}$ graphs. It suffices to go through all vertices v and determine the best solution including v. Such a solution consists of the best solution which is compatible with v, i.e. the optimal solution of KCG for capacity $c - w(v)$ on $G[V \setminus N[v]]$, which can be computed in pseudopolynomial time by assumption.

As an example for the above argument, it follows immediately from Pferschy and Schauer [32] that KCG allows a pseudopolynomial algorithm and also an FPTAS for *nearly chordal* graphs.

The next definition is a variant of the more general definition that can be found in Pruhs and Woeginger [34].

Definition. *Subset Selection Problem*

A subset selection problem is defined by a ground set X with n elements each of which has associated a positive profit $p(x)$ for $x \in X$ and for each subset Y of X it can be decided in polynomial time whether Y is feasible. Moreover assume that every instance of the subset selection problem has a feasible solution. Then we are looking for a feasible subset of X with maximum total profit.

The following general result about the existence of an FPTAS based on a given pseudopolynomial algorithm will be applied frequently throughout this paper.

Theorem 1 (Pruhs and Woeginger [34])**.** *If there exists an exact algorithm for a subset selection problem with running time polynomial in n and in $\sum_{x \in X} p(x)$ then there exists also an FPTAS for this problem.*

It is easy to see that KCG belongs to this family of subset selection problems. Note that it is trivial to find a feasible solution for KCG (take the empty set or any single item as a solution). However, for KFG on a general forcing graph it is strongly $\mathcal{NP}$-complete to determine even the existence of a feasible solution: Indeed, given an arbitrary instance of the minimum vertex cover problem, one could define an instance of KFG on the same graph with all item weights equal to 1. Then there exists a vertex cover of size at most c iff KFG with capacity c has a feasible solution.

However, the vertex cover problem is polynomially solvable on all graph classes for which we apply Theorem 1 in this paper. Thus, one can check the feasibility of an instance in a preprocessing step and restrict KFG to instances with a feasible solution.

2. Weakly chordal conflict graphs

The main goal of this section is to derive an FPTAS for KCG on weakly chordal graphs, which are a well studied superclass of chordal graphs (e.g. Cameron et al. [12], Hayward et al. [21]). It is known that weakly chordal graphs are perfect.

A *hole* is an induced cycle with 5 (i.e. a C_5) or more vertices and an *anti-hole* is the complement of a hole. A graph is *chordal* (also known as *triangulated graph*) if it contain neither induced holes nor a C_4. In other words, any cycle with more than 3 vertices must contain a chord. A graph is *weakly chordal* if it does not contain induced holes and anti-holes.

The weighted independent set problem was polynomially solved on weakly chordal graphs by Hayward et al. [20]. Their algorithm was later improved by Spinrad and Sritharan [36]. However these algorithms cannot be adopted to solve the knapsack problem with conflicts: They perform a sequence of iterations to transform the given weakly chordal graph into intermediate graphs, which are all weakly chordal, until finally a clique is obtained. In every transformation step the weight of the maximum weight independent set of the involved graphs stays the same. Unfortunately the algorithms modify non-optimal independent sets which might represent optimal solution sets in the disjunctively constrained knapsack problem.

As usual we define a *triangulation* of a graph $G = (V, E)$ as a chordal supergraph $G' = (V, E \cup E')$ of G, where the edges from E' are called *fill edges*. A minimal triangulation of G is a triangulation G' such that removing any edge from E' results in a non-chordal graph. If there exists a minimal triangulation G' such that $V' \subseteq V$ induces a maximal clique in G', then V' is called a *potential maximal clique*.

Bouchitté and Todinca [5] proved that all potential maximal cliques of a graph can be generated in time that is polynomial in the number of minimal separators of a graph. Moreover Bouchitté and Todinca [4] proved that weakly chordal graphs contain polynomially many minimal separators. Combining these results leads to the well known result that weakly chordal graph contain only a polynomial number of potential maximal cliques.

Fomin and Villanger [17] gave an algorithm that finds large induced subgraphs of bounded treewidth of a graph G which contains only polynomially many potential maximal cliques. Noting that an independent set has treewidth 0, Lokshtanov et al. [25] showed that the algorithm given by Fomin and Villanger [17] can be modified in order to solve the weighted independent set problem.

Theorem 2. *Lokshtanov et al. [25, Proposition 1] There is an algorithm that given as input a vertex weighted graph G on n vertices and m edges, together with a list Π of potential maximal cliques, outputs in time $O(|\Pi|n^5 m)$ the weight of the maximum weight independent set I such that there exists a minimal triangulation H of G such that every maximal clique C of H is on the list Π and satisfies $|C \cap I| \leq 1$.*

Fomin and Villanger [17] use minimal triangulations in their dynamic program, whereas [25] use a special tree-decomposition which is related to the clique tree of a minimal triangulation. We recapitulate their main definitions and technical prerequisites: let I be an independent set in G. An *I-good minimal triangulation* is a minimal triangulation of G such that no vertex of I is incident to a fill edge. A tree-decomposition $(T, \mathcal{X})$ of G is called *I-sparse* if each bag $B \in \mathcal{X}$ contains at most one vertex from I (i.e. $|I \cap B| \leq 1$). A rooted tree-decomposition is *simple* if no bag B is a subset of any other bag B' and for every $u, v \in V(T)$ where v is a descendant of u in T, there exists a component $C \in C(\mathcal{X}(u))$ such that $\mathcal{X}(v) \subset \mathcal{X}(u) \cup C$.

The connection between minimal triangulations and simple tree-decompositions was established by [25]. They proved that every minimal triangulation has a clique tree that is a simple tree-decomposition and on the other hand that any simple tree-decomposition of G, whose bags are potential maximal cliques of G, is a clique tree of a minimal triangulation H of G.

Moreover, they proved that for every independent set I in G there exists a minimal triangulation which is I-good. With the above statement it follows that for such an I-good minimal triangulation there always exists an I-sparse simple tree-decomposition since all bags of the tree-decomposition are maximal cliques of the triangulation. Thus, the following Theorem 3 (quoted from the original source) outputs an independent set I which solves the weighted independent set problem on G if a complete list of all potential maximal cliques is given.

Theorem 3. *Lokshtanov et al. [25, Lemma 16] There is an algorithm that given as input a vertex weighted graph G on n vertices and m edges, together with a list Π of potential maximal cliques of G, outputs in time $O(|\Pi|^2 n^4 m)$ the weight of the maximum weight independent set I such that there exists an I-sparse simple tree-decomposition $(T, \mathcal{X})$ of G such that $\mathcal{X}(v) \in \Pi$ for all $v \in V(T)$. If no such independent set exists, the algorithm outputs $-\infty$.*

Note that due to the different techniques used, Theorem 2 provides a better running time than the algorithm of Theorem 3. Due to the more accessible technique we decided to follow the approach presented in Lokshtanov et al. [25] and solve KCG by dynamic programming over tree-decompositions on all graph classes whose potential maximal cliques can be listed in polynomial time.

Let B be a potential maximal clique from the set Π of all potential maximal cliques. We look at B as a graph separator and denote by $\mathcal{C}_B = \{C_B^1, \ldots, C_B^{k_B}\}$ the set of all connected components of $G \setminus B$. The indices $1, \ldots, k_B$ are assigned according to the size of the components with $|C_B^i| \leq |C_B^{i+1}|$ and ties are broken arbitrarily. Let $K_B^j = \bigcup_{i=1}^{j} C_B^i$ denote the partial union of components. A restriction of the set of components to those which are fully contained in some given set $X \subseteq V$ is denoted by $\mathcal{C}_B(X) = \{C \in \mathcal{C}_B \mid C \subseteq X\}$. The elements of $\mathcal{C}_B(X)$ are ordered according to indices given to the elements of $\mathcal{C}_B$: If $\mathcal{C}_B(X) = \{C_B^1(X), C_B^2(X), \ldots, C_B^{k_B(X)}(X)\}$, this means that two sets $C_B^i(X)$, $C_B^{i+1}(X)$ correspond to sets $C_B^{i'}$ and $C_B^{j'}$ from $\mathcal{C}_B$ with $i' < j'$. Note that this ordering is not strictly necessary for the correctness of the described method but avoids recursive evaluations. Again, the partial union is defined as $K_B^j(X) = \bigcup_{i=1}^{j} C_B^i(X)$.

In the following we proceed along the lines and notation layed out in Lokshtanov et al. [25]. However, our approach is more complicated since it does not suffice to collect "large" independent sets from subgraphs but the weight and profit values of any possible subset of the solution has to be kept explicitly for further consideration.

We will introduce a dynamic programming function $M = M(B, S, K_B^j(X), p)$ which takes as arguments a potential maximal clique $B \in \Pi$, a set $S \subseteq B$ which contains either a single vertex of B or is empty (S represents the potential inclusion of the associated item into the solution), a partial union $K_B^j(X)$ of components separated by B according to the above definition for some restricting set $X \subseteq V$, and finally a profit value $p \in \{0, 1, 2, \dots, P\}$. M is now a function that returns the weight of the minimum weight independent set $I \subseteq B \cup K_B^j(X)$ with a profit of $p + p(S)$ such that $I \cap B = S$ and the following property of a tree-decomposition: There exists an I-sparse simple tree-decomposition $(T, \mathcal{X})$ of $G[B \cup K_B^j(X)]$ such that all bags of $(T, \mathcal{X})$ are potential maximal cliques and B is the root vertex of T. If no such I exists, then M returns ∞.

Now we construct a recursion to determine the values of M as follows. For a certain potential maximal clique $B \in \Pi$, a single candidate vertex represented by S and some restricting set X we consider the components $\mathcal{C}_B(X)$ resulting from the separation by B in increasing order of their size and add them one after the other. In the iteration appending component $C_B^j(X)$ we split the target profit value p into a part p' contributed by vertices (compatible with S) from previous components $C_B^1(X), \dots, C_B^{j-1}(X)$ which is already summarized in $K_B^{j-1}(X)$ (see (6)) and the complementing part $p - p'$ contributed by vertices in $C_B^j(X)$. So we can split the independent set I corresponding to the optimal value of $M(B, S, K_B^j(X), p)$ into parts $I_1, \dots, I_j$ where $I_i := (I \cap C_B^i(X)) \setminus S$.

The difficult part arises from considering the connection between $C_B^j(X)$ and the separator B. To allow a recursive evaluation we have to separate also $C_B^j(X)$. To do so, we go through all potential maximal cliques $B' \in \Pi$ which may serve as a "connector" between B and the components of $C_B^j(X)$ arising from a separation by B' (see (7)). At the same time the inclusion of the vertex represented by S has to be kept feasible also w.r.t. $C_B^j(X)$ and the existence of the required tree-decomposition has to be kept alive.

This leads to a minimization over all potential maximal cliques $B' \in \Pi$ and all subsets S' fulfilling the following properties: (a) $B' \subseteq B \cup C_B^j(X)$, (b) $N(C_B^j(X)) \subseteq B'$, (c) $B' \cap C_B^j(X) \neq \emptyset$, (d) $|S'| \leq 1$, (e) $B \cap B' \cap S = B \cap B' \cap S'$, (f) $G[S \cup S']$ is independent.
(a)–(c) imply that B' serves as a "connector" between B and $C_B^j(X)$ whereas (e) and (f) state the compatibility of S and S'. In fact, for $|S| = 1$ they only allow the following two cases: Either $S = S'$ or (if both are non-empty) S is not in B', S' is not in B and the corresponding vertices are not adjacent.

With these preliminaries we now state the crucial recursion[1].

$$
\begin{aligned}
M\left(B,S,K_B^j(X),p\right) = w(S) + \min_{p'\le p}\Bigg\{ & M\left(B,S,K_B^{j-1}(X),p'\right) - w(S) + && (6)\\
& + \min_{\{B',S' \mid (p-p'-p(S'\setminus S))\ge 0\}} \Big\{ w(S'\setminus S) + && (7)\\
& + M\left(B',S',K_{B'}^{k_{B'}(\cdot)}(C_B^j(X)),p-p'-p(S'\setminus S)\right) - w(S')\Big\}\Bigg\} && (8)\\
& \text{where } B',S' \text{ fulfill (a)–(f)}
\end{aligned}
$$

The initialization for the first component, i.e. $K_B^1(X) = C_B^1(X)$, is defined in the following way:

$$
\begin{aligned}
M\left(B,S,K_B^1(X),p\right) = w(S) + \min_{\{B',S' \mid (p-p(S'\setminus S))\ge 0\}} \Big\{ & w(S'\setminus S) + && (9)\\
& + M\left(B',S',K_{B'}^{k_{B'}(\cdot)}(C_B^1(X)),p-p(S'\setminus S)\right) - w(S')\Big\}\Big\}\\
& \text{where } B',S' \text{ fulfill (a)–(f)}
\end{aligned}
$$

Lemma 4. *Recursion (6)-(8) with initialization (9) correctly computes the values of $M(B,S,K_B^j(X),p)$ for any given $X \subseteq V$ and $p = 0,1,\ldots,P$.*

Proof.

" $\le$ "

For the three parameters p', B' and S' that minimize the right hand side of (6) – (8), I_j is the independent set of $G[C_B^j(X) \cup B']$ consuming a weight of

$$M\left(B',S',K_{B'}^{k_{B'}(\cdot)}(C_B^j(X)),p-p'-p(S'\setminus S)\right)$$

and I' the independent set of $G[B \cup K_B^{j-1}(X)]$ consuming a weight of

$$M\left(B,S,K_B^{j-1}(X),p'\right)$$

(if they exist). If one of these weight values is ∞ we are done. We now have to show that $I := I_j \cup I'$ is independent and that there exists an I-sparse simple tree-decomposition $(T,\mathcal{X})$ of $G[B \cup K_B^j(X)]$ such that all bags of $(T,\mathcal{X})$ are potential maximal cliques and B is the root vertex of T. All pairs of vertices from $K_B^{j-1}(X)$ and $C_B^j(X)$ are independent since they are separated by vertices from B. If I_j contains a vertex from B then by property (e) it must be the vertex in S, hence I is independent. We also know that there exists an I_j-sparse simple

[1]The number of components $k_B(X)$ in $\mathcal{C}_B(X)$ for some X will be abbreviated by $k_B(\cdot)$.

tree-decomposition $(T_j, \mathcal{X}_j)$ of $G[C_B^j(X) \cup B']$ such that all bags of $(T_j, \mathcal{X}_j)$ are potential maximal cliques and B' is the root vertex of T_j. There also exists an I'-sparse simple tree-decomposition $(T', \mathcal{X}')$ of $G[B \cup K_B^{j-1}(X)]$ such that all bags of $(T', \mathcal{X}')$ are potential maximal cliques and B is the root vertex of T. We will now merge these two decompositions into a simple tree-decomposition $(T, \mathcal{X})$ of $G[B \cup K_B^j(X)]$, where $\mathcal{X} = \mathcal{X}' \cup \mathcal{X}_j$: the new tree structure T results from adding the edge (B, B') to the disjoint union of T_j and T'. $(T, \mathcal{X})$ is a tree decomposition because of property (b). It is I-sparse because of the respective sparseness of $(T', \mathcal{X}')$ and $(T_j, \mathcal{X}_j)$ and property (e). Moreover, it is simple because both tree-decompositions are simple and property (a) and (c). Hence $M\left(B, S, K_B^j(X), p\right) \leq p(I)$.

" $\geq$ "

Let I be an independent set of $G[B \cup K_B^j(X)]$ of weight $M\left(B, S, K_B^j(X), p\right)$ such that $I \cap B = S$ and $(T, \mathcal{X})$ is an I-sparse simple tree-decomposition of $G[B \cup K_B^j(X)]$ with root bag B, i.e. $\mathcal{X}(r(T)) = B$. For $C_B^j(X)$ there is exactly one child vertex $V_j \in T$ of $r(T)$ with $\mathcal{X}(V_j) \cap C_B^j(X) \neq \emptyset$: if there exists another child V_j' of $r(T)$ such that $\mathcal{X}(V_j')$ has a nonempty intersection with $C_B^j(X)$ we get that also B has to contain an element of $C_B^j(X)$: since $(T, \mathcal{X})$ is a tree-decomposition all vertices of bags of the sub tree-decomposition rooted in V_j are separated by B from all vertices of bags of the sub tree-decomposition rooted in V_j'.

Denote $\mathcal{X}(V_j)$ as B' and $I \cap B'$ as S'. These two sets obviously satisfy conditions (a) and (c)–(f). Assume that (b) is not satisfied and hence B contains an element v from $N(C_B^j(X))$ which is not in B'. Let $w \in C_B^j(X)$ be a neighbor of v. Denote by $U_1, \ldots, U_k$ all child vertices of $V_j \in T$. We know that v is not in $\mathcal{X}(U_i)$ for any i or in any of its childs since this would imply that it is also in B. Moreover, w is not in B, but this already gives a contradiction since there has to be a bag in T containing u and w.

Let $(T_j, \mathcal{X}_j)$ be the sub tree-decomposition of $(T, \mathcal{X})$ rooted in B' and $(T', \mathcal{X}')$ the tree-decomposition which results from $(T, \mathcal{X})$ by removing $(T_j, \mathcal{X}_j)$. Let $I_j = I \cap (B' \cup C_B^j(X))$ and $\bar{I} = I \setminus I_j$. Both $(T_j, \mathcal{X}_j)$ and $(T', \mathcal{X}')$ are simple because of the simplicity of $(T, \mathcal{X})$ and both are sparse because of the I-sparsity of $(T, \mathcal{X})$. Hence, $(T_j, \mathcal{X}_j)$ is an I_j-sparse simple tree-decomposition and $(T', \mathcal{X}')$ is an $\bar{I}$-sparse simple tree decomposition. It follows that I_j and $\bar{I}$ can be chosen as independent sets on the left hand side of the equation and thus the " $\geq$ " side of the equality follows. □

Theorem 5. *Given the complete list* Π *of all potential maximal cliques of* G*, KCG can be solved to optimality in time*

$$O(|\Pi|^3 \cdot n^4 \cdot P^2).$$

Proof. It follows from Lemma 4 that by evaluating function M we can determine an independent set I of maximum total profit and total weight $\leq c$ such that there exists an I-sparse simple tree-decomposition $(T, \mathcal{X})$ of G such that $\mathcal{X}(v) \in \Pi$ for all $v \in V(T)$. As pointed out before, this is equivalent to finding the optimal solution of KCG. It can be computed by taking among all potential maximal cliques B and subsets S of cardinality ≤ 1 the maximum profit p^* such that:

$$M(B, S, K_B^{k_B}(V), p^* - p(S)) \leq c$$

To achieve the stated running time complexity we apply a preprocessing step and calculate for all possible potential maximum cliques $B \in \Pi$ all the $O(n)$ components of $\mathcal{C}(B)$, which can be done in $O(|\Pi|(m+n))$ time by breadth first search. In a next step we check for all pairs of components whether one of them is a subset of the other and store the result in a containment matrix of size $O(|\Pi|^2 n^2)$. Since each check can be done in linear time this takes all together $O(|\Pi|^2 n^3)$ time. In the following we can use this information to check in constant time if a component is a subcomponent of another one.

Let us estimate the number of different argument tuples for M: There are $|\Pi|$ different potential maximal cliques B each of them containing at most n subsets S with one item. The partial union $K_B^j(X)$ is taken over $k_B(X) \leq n$ different values of j and each of them might be restricted by any of the $|\Pi|n$ different components X. Multiplying these choices the number of different argument tuples for M can be roughly bounded by

$$O(|\Pi| \cdot n \cdot n \cdot |\Pi| n \cdot P).$$

The function M is now evaluated for each of these possible argument tuples. This is done by first sorting all $|\Pi|n$ possible components in increasing order of size. Clearly, a component of minimum size cannot contain any of the other components properly. Hence, if a potential maximal clique B induces more than one, say i, components $\in \mathcal{C}_B$ of minimum size we can easily compute $M(B, S, K_B^i(C'), p)$ for all possible choices of C'. If we now go to components of second smallest size we can compute all relevant tuples of M relying on the values calculated in the previous step. This process continues until we reach the components of largest size.

Performing an evaluation of M for a certain tuple requires $O(|\Pi| \cdot n \cdot P)$ values of previously calculated values of M. Hence we get a total running time of

$$O(|\Pi| \cdot n \cdot n \cdot |\Pi| n \cdot P \cdot |\Pi| \cdot n \cdot P) = O(|\Pi|^3 \cdot n^4 \cdot P^2)$$

□

Remark 1. The sorting of the components by their size is not really necessary. Alternatively, one could start the recursion $M(B, S, K_B^{k_B(V)}(V), c - w(S))$ for all choices of B and S and store values of M with certain argument tuples when they are actually computed for the first time during a recursion. These stored values can then be used in other recursive calls in order to avoid calculating the function with the same tuple set more than once.

By Bouchitté and Todinca [4] weakly chordal graphs have $O(n^2)$ minimal separators which can be computed in $O(n^5)$ time by Berry et al. [1]. By Bouchitté and Todinca [5] the potential maximal cliques of any graph can be listed in $O(n^2 m|\Delta_G|^2)$, where Δ_G is the set of all minimal separators of G. Hence we get the following corollary:

Corollary 6. *KCG can be solved in $O(n^{22} \cdot m^4 \cdot P^2)$ time on weakly chordal graphs.*

It should be clear that this is only a very loose upper bound on the running time, but its improvement is not in the scope of this paper. By Theorem 1 we can state the main result of this section:

Corollary 7. *There is an FPTAS for KCG on weakly chordal graphs.*

Remark 2. The main result of Lokshtanov et al. [25] was giving a polynomial time algorithm for the weighted independent set problem in P_5-free graphs. They showed that for this graph class a subset of all potential maximum cliques of polynomial size can be found in polynomial time that allows to find the maximum weight independent set. Note that the set of potential maximum cliques found by them also guarantees to find the optimal KCG solution when plugged into our algorithm.

We conclude that there exists an FPTAS for KCG on P_5-free graphs.

3. Modular Decomposition and Clique Separators

In this section we consider graph decomposition techniques that were widely used in the literature for solving the (weighted) independent set problem on special graph classes.

A *clique separator* C of a connected graph G is a separator (recall Section 1.1) which is a clique. By definition of a separator the graph $G[V \setminus C]$ has ℓ connected components $A_1, A_2, \ldots, A_\ell$ with $\ell \geq 2$.

A *clique separator decomposition* recursively separates $G_1 = G[A_1 \cup C]$, $G_2 = G[A_2 \cup C], \ldots, G_\ell = G[A_\ell \cup C]$ by clique separators. Any subgraph not containing a separating clique is called *atom*. Following Tarjan [37] the decomposition procedure can be represented by a clique decomposition tree, where each inner node represents a clique separator (as a set of vertices) and all leaves represent atoms (again as sets of vertices). It is important to note that the decomposition algorithm presented by Tarjan [37] separates an atom in every step, i.e. in the resulting decomposition tree all inner nodes lie on a path.

Clique separator decomposition was applied frequently in the literature to solve $\mathcal{NP}$-hard problems on special graph classes. We will mention only the classical results by Tarjan [37] and Whitesides [38] for maximum clique and independent set on special graph classes such as chordal graphs, clique separable graphs and EPT-graphs.

A *module* of a graph G is a set of vertices M such that every vertex outside M is either connected to all vertices in M or to none of them. Informally speaking, all vertices in M "look the same" for vertices in $G \setminus M$. Formally, for all $v \in G \setminus M$ either $N(v) \cap M = M$ or $N(v) \cap M = \emptyset$. A module is non trivial if $1 < |M| < |V|$.

The concept of modular decomposition was introduced by Gallai [18] and appears in the literature under different names, e.g. substitution decomposition in Möhring and Rademacher [29]. It was applied frequently for the solution of $\mathcal{NP}$-hard problems on special graph classes, e.g. by Lozin and Milanič [26] for the solution of the maximum weight independent set on fork-free graphs.

Following Brandstädt and Hoàng [8] a graph with non trivial module M can be decomposed in the following way: $G_1 = G[M]$ and $G_2 = G[(V \setminus M) \cup \{v_M\}]$ where v_M is a new vertex which has the same adjacencies in $G[V \setminus M]$ as the module M. This means that G_1 is the subgraph of G induced by M and G_2 is a subgraph of G where the module M is removed and replaced by a single vertex v_M. From a computational point of view, it makes sense to use only *maximum* modules. If this process is applied recursively one gets a *modular decomposition* of the graph which can be represented by a decomposition tree $\mathcal{M}$. Note that $\mathcal{M}$ is a full binary tree, i.e. a tree where every node (with exception of the leaves) has exactly two child nodes. The leaves of $\mathcal{M}$ correspond to subgraphs that cannot be further decomposed by non trivial modules. These graphs are called *prime*.

In the literature a slightly different decomposition process is frequently applied (cf. McConnell and Spinrad [27]), where G is first partitioned into connected and co-connected components. Each of the resulting subgraphs can be partitioned into modules. However, to preserve the analogy to the clique decomposition we stick to the version described above with one module contracted in each step.

In this section we follow the approach of Brandstädt and Giakoumakis [7] and apply both modular and clique decompositions. This means that we start with a modular decomposition which yields a collection of prime graphs. Then we apply clique decomposition to each of these prime graphs until we reach the atoms of these graphs. Note that the resulting atoms are not necessarily prime. The main contribution of this section is the following result.

Theorem 8. *If KCG can be solved in pseudopolynomial time on the atoms of the prime graphs resulting from a modular decomposition of G, then KCG can also be solved in pseudopolynomial time on G.*

Based on this theorem we can exploit results from the literature of the following type:

(i) Given a graph G with some property $\mathcal{P}$, the modular decomposition yields prime graphs inheriting property $\mathcal{P}$.

(ii) Given a prime graph with property $\mathcal{P}$, its atoms have property $\mathcal{Q}$.

Now Theorem 8 states that if KCG can be solved in pseudopolynomial time on graphs with property $\mathcal{Q}$, then KCG can be solved in pseudopolynomial time also on graphs with property $\mathcal{P}$. The relevance of this statement lies in the fact that in many cases atoms of prime graphs allow much more restrictive properties than the original graph and thus pseudopolynomial results requiring a more restrictive property $\mathcal{Q}$ can be extended to a wider family of graphs observing only a weaker property $\mathcal{P}$. At the end of this section we will give a number of examples for graph properties $\mathcal{P}$ which allow a pseudopolynomial solution of KCG through an application of Theorem 8. We will assume that any given pseudopolynomial algorithm returns not only a single optimal solution value but a full array of weight values for every target profit smaller than the given upper bound[2].

In the following we will present a dynamic programming approach for KCG. During the decomposition approach subproblems will be solved and their optimal solutions contracted into single vertices. Therefore, we introduce the best solution obtained from including a single vertex v. This includes item v and possibly the solutions of subproblems previously merged into v. For every profit value p, $0 \leq p \leq P$, let $d_p(v)$ be the *minimum weight* required for a feasible solution given by item v with total profit p. As an initialization we use $d_{p_v}(v) = w_v$ and $d_p(v) = \infty$ (or a large constant) for $p \neq p_v$.

With this definition we only work with function values of d and never refer explicitly to profits and weights of individual items. In the beginning, $d(v)$ contains only the feasible solution $d_{p_v}(v) = w_v$ corresponding to the singleton solution $\{v\}$. In the course of the computation more extensive solutions will be determined and their profits and weights inserted into $d(v)$. The corresponding subgraphs of G will then be deleted and their information remains attached to v via $d(v)$.

For a subset of vertices (items) $V' \subseteq V$ we will also consider the optimal solution of KCG on $G[V']$ derived either by recursion or by a direct algorithm available for graphs with a certain property. The corresponding solution values will be represented by a dynamic programming array, where $z_p(V')$ gives the *minimum weight* of a feasible solution with profit p for KCG restricted to $G[V']$. It is necessary to define $z_0(V') = 0$ as an empty solution.

[2]If this is not the case, the algorithm could be performed iteratively thus increasing the pseudopolynomial running time by a factor of P.

3.1. Modular decomposition

Recall that every inner node of the decomposition tree $\mathcal{M}$ represents a subgraph $G[V']$ and has two child nodes. One of them corresponds to a module M of $G[V']$, while the other represents a copy of $G[V']$ with M contracted to a single vertex v_M. Now we first consider the node of module M and solve KCG on $G[M]$ recursively. This returns an array $z_p(M)$ which we assign to the vertex v_M in the second child node as $d_p(v_M)$. Note that by definition of a module it does not matter which items from M are included in a solution represented by $z_p(M)$. Any solution in $G[V' \setminus M]$ which is not in conflict with the new vertex v_M can also be combined with any solution for M.

Then we proceed recursively and let this second child node take over the role of the inner node, i.e. $V' := V' \setminus M \cup \{v_M\}$. At the end of this recursive process we are left with a prime graph for which we can solve KCG by assumption.

Note that this prime graph is the unique leaf of the modular decomposition tree which is reached from the root by a path that never moves to a child node representing a module. Thus, the solution for this final leaf contains the information of all modules separated before and constitutes the solution of KCG on the original graph G.

3.2. Clique Decomposition

Given a graph G, we follow the approach described by Tarjan [37] for the maximal weight independent set problem and find a clique separator C which separates G into components A and B such that $G[A \cup C]$ is an atom. In fact the decomposition procedure described by [37, Sec. 2] separates an atom in each step and runs in $O(nm)$ time. Analogous to the modular decomposition, we first consider the atom $G[A \cup C]$. Later, $G[B \cup C]$ will be considered recursively.

Each vertex $q \in C$ might be selected in a solution. If this is the case, it can be complemented by the best possible solution in $G[A]$, which is not in conflict with q, i.e. the best solution in $G[A \setminus N(q)]$. The profit and weight information of the resulting solutions containing q is then inserted into $d_p(q)$. To include also the case that no vertex of C is selected, we add an auxiliary *empty* vertex e into C with $N(e) = C$, $d_0(e) = 0$ and $d_p(e) = \infty$ for $p > 0$. Formally, we have for every $q \in C$:

$$d_p(q) := \min_{p_1,p_2} \{d_{p_1}(q) + z_{p_2}(A \setminus N(q)) \mid p_1 + p_2 = p\} \text{ for } p = 0, \dots, P \qquad (10)$$

This equation determines the best, i.e. lowest weight solution for every possible profit value p by combining q contributing profit p_1 with a compatible subset of A with profit p_2.

By assumption, KCG can be solved for any atom of G in pseudopolynomial time. To evaluate (10) we also have to solve KCG on $G[A \setminus N(q)]$, i.e. on the subgraph of an atom arising from eliminating a single vertex and its neighborhood. It is easy to see that the pseudopolynomial algorithm applies also to these subgraphs: In most cases property $\mathcal{Q}$ will also be valid for all subgraphs of an atom and we are done. But even if this is not the case one can easily enforce the inclusion of q by temporarily setting the weight of q to 0 and its profit to P. Then we run the algorithm on the full atom, but replace each target profit p by $p + P$. Trivially, q will be included in the solution and it is complemented by the best solution of KCG on the desired subgraph with profit p.

After completion of (10) all potential contributions of items in A to an overall solution are represented by array $d_p(q)$ for all $q \in C$. Since C is a clique, these can contribute to the overall solution at most once.

Then we proceed recursively and search for a clique separating an atom from $G[B \cup C]$. This atom is treated as above with dynamic programming entries computed for each vertex in the separating clique.

Continuing this process, we finally obtain a graph which is separated into two atoms both of which can be resolved and their contribution contracted in the final separator. It remains to pick the vertex from this separator that delivers the best solution value.

It would be interesting to combine modular and clique decomposition in an arbitrary order and thus generalize the above setting to prime atoms instead of atoms of prime graphs. Indeed, this would make the dynamic programming evaluation much more complicated and we followed several steps in this direction in the spirit of Brandstädt and Hoàng [8, Theorem 8]. Unfortunately, it was stated in the recent paper Brandstädt and Giakoumakis [7] that the approach implied by [8, Corollary 9] does not seem to work and only the iterative process of considering atoms of a prime graph can be pursued.

3.3. Applications

It is clear from the construction that the clique decomposition yields $O(n)$ atoms for a graph with n vertices since each decomposition step for some clique C separates at least one vertex. To solve (10) in each such step a pseudopolynomial algorithm for atoms has to be executed $|C| + 1$ times (i.e. $O(n)$) and a scan through all profit values has to be performed $|C|$ times taking $O(nP)$ time. Furthermore, the decomposition tree $\mathcal{M}$ of the modular decomposition has $O(n)$ leaves each of which requiring the execution of a pseudopolynomial algorithm for prime graphs. Without caring about the details of the running time, it follows that any pseudopolynomial algorithm for the atoms of prime graphs gives rise to a pseudopolynomial algorithm for the original graph as stated in Theorem 8.

We will now describe a few graph classes for which Theorem 8 implies a pseudopolynomial time algorithm for KCG based on properties $\mathcal{P}$ and $\mathcal{Q}$. Clearly, there are many more graph classes around where our framework would apply but we restrict ourselves to some more recent examples.

A *co-chair* is a graph with five vertices $a, \dots, e$ and six edges (a, b), (a, c), (b, c), (b, d), (c, d), (d, e). Brandstädt and Giakoumakis [6, Sec.3] give some examples for relevant classes of (hole, co-chair)-free graphs. Note that some members of this family are perfect graphs while others are not. It was shown in Brandstädt and Giakoumakis [7] that atoms of prime (hole, co-chair)-free graphs are nearly weakly chordal ($\mathcal{Q}$). Since the modular decomposition of (hole, co-chair)-free graphs trivially yields (hole, co-chair)-free prime graphs ($\mathcal{P}$), we get from Corollary 7:

Corollary 9. *There is an FPTAS for KCG on (hole, co-chair)-free graphs.*

Given a co-chair, a *paraglider* arises from adding the additional edge (a, e). Brandstädt et al. [11, Sec.1] discuss the relevance of (hole, paraglider)-free graphs which are perfect graphs and contain chordal graphs as a subclass. They present a decomposition result [11, Theorem 1] which states that the atoms of (hole, paraglider)-free graphs belong to one of the following three graph families (in fact this property is necessary and sufficient for (hole, paraglider)-free graphs). Each of them allows a simple pseudopolynomial algorithm for *KCG*:
(i) *Complete multipartite graph* allow a partitioning of the vertex set into k subsets such that two vertices are adjacent if and only if they belong to different subsets. Clearly, any solution of *KCG* can contain only vertices from exactly one subset. It suffices to solve a standard knapsack problem for each subset and select the best of the k resulting solutions.
(ii) *Join of a chordal bipartite graph and a clique*, where the former is a bipartite, hole-free graph. Since chordal bipartite graphs are exactly those bipartite graphs, which are weakly chordal, the solution of *KCG* follows from Section 2 by choosing between the optimal solution on the chordal bipartite graph and the best vertex of the clique.
(iii) *Join of a matched co-bipartite graph and a clique*, where the former consists of two disjoint cliques of size k and the edges between the cliques form a matching with k edges. Obviously, *KCG* can be solved on such a graph by taking each vertex of one clique as a candidate and combine it with the best vertex (taking the weight constraint into account) of the other clique excluding the matching partner of the candidate.

A *diamond* is a K_4 with one missing edge. The structure of (hole, diamond)-free graphs, which generalize chordal bipartite graphs, was described in Berry et al. [2]. They showed that an atom of a (hole, diamond)-free graph is either a clique, or a chordal bipartite graph, or a matched co-bipartite graph. These properties are very similar to the characterization of atoms of (hole, paraglider)-free graphs given above and *KCG* can be solved in the same way. Summarizing, we have:

Corollary 10. *There is an FPTAS for KCG on (hole, diamond)-free graphs and on (hole, paraglider)-free graphs.*

More special graph classes defined by certain forbidden subgraphs were described in Brandstädt et al. [10] (with the correction mentioned in [7]). For some of them it could be shown that atoms of prime graphs resulting from modular decomposition of the given graph are e.g. nearly chordal and thus an FPTAS for KCG exists.

4. KCG on H-minor free conflict graphs

In this section we consider KCG restricted to a H-minor free conflict graph. A a graph H is a *minor* of G if H can be obtained by successively applying the following three operations on G: deleting isolated vertices, deleting edges and contracting edges. Moreover a graph class is called *H-minor free* if the graph H is not a minor of any of the graphs of this class. The most well-known H-minor free graph class are planar graphs.

By the strong $\mathcal{NP}$-hardness of the independent set problem on cubic planar graphs we immediately get that there cannot exist an FPTAS for KCG on H-minor free graphs. Complementing this negative result we will now show that there exists a PTAS for KCG on planar graphs. We will indeed more generally show that a result of Demaine et al. [14] extends to a class of problems that includes KCG on H-minor free graphs as a special case.

If a graph property π valid for a graph G also holds for all of its subgraphs, then π is called hereditary. [14] defined the maximum cardinality (profit) induced subgraph problem for a graph property π ($MISP(\pi)$) as a maximum cardinality (resp. profit) subset of vertices S of a graph G such that the hereditary property π holds for the subgraph induced by S. They defined $EMISP(\pi)$ to be the problem of finding the maximum cardinality (profit) set of edges S' which induces a subgraph that fullfills the hereditary property π.

Theorem 11. *Demaine et al. [14, Theorem 3.7] For any hereditary graph property π that can be solved in polynomial time on graphs of bounded treewidth, for any graph H, and for any $\varepsilon > 0$, there is a polynomial-time $(1+\varepsilon)$-approximation algorithm for $MISP(\pi)$ and $EMISP(\pi)$ on H-minor-free graphs.*

An important decomposition result for H-minor free graphs is also shown in [14].

Theorem 12. *Demaine et al. [14, Theorem 3.1] For a fixed graph H, there is a constant c_H such that, for any integer $k \geq 2$ and for every H-*minor-free graph *G, the vertices of G can be partitioned into k sets such that any $k-1$ of the sets induce a graph of treewidth at most $c_H k$. Furthermore, such a partition can be found in polynomial time.*

In the following we consider a generalization of KCG. Let c be a knapsack capacity. We say that a graph G with vertex (resp. edge) weights $w(v)$ (resp. $w(e)$) fullfils c if $\sum_{v\in V} w(v) \leq c$ (resp. $\sum_{e\in E} w(e) \leq c$). Let $p(v)$ and (resp.

$p(e)$) be the profit of vertex v (resp. edge e). With this we define the *maximum profit induced subgraph problem* for a graph property π and a capacity c ($MPISP(\pi, c)$) as finding a maximum profit subset S of vertices of a graph G such that the subgraph induced by S fullfils π and c. Similarly, we define $EPISP(\pi, c)$ as the problem of looking for the maximum profit *edge* set S' of G that induces a subgraph fulfilling π and c.

By applying Theorem 12 we get the following result which is an extension of Theorem 11.

Theorem 13. *Let π be a hereditary graph property and c a knapsack constraint. Let H be a fixed graph. If the problem $MPISP(\pi, c)$ (resp. $EPISP(\pi, c)$) can be approximated by an $(1+\varepsilon)$ polynomial time approximation algorithm on graphs of bounded treewidth (for all fixed $\varepsilon > 0$) we get:*

- *For any fixed $\delta > 0$, there is a polynomial time $(1+\delta)$-approximation algorithm for $MPISP(\pi, c)$ and $EPISP(\pi, c)$ on H-minor-free graphs.*

Proof. We first concentrate on $MPISP(\pi, c)$:

By Theorem 12 we get disjoint subsets $V_1, \ldots, V_k$ such that any $k-1$ of them induce a graph of bounded treewidth. For each $i \in \{1, \ldots, k\}$ we solve the $(1+\varepsilon)$-approximation algorithm on the graph induced by $\bigcup_{j \neq i} V_j$ and take the best over all these solutions with profit value z^a and corresponding solution set of vertices S^a. Note that the exact value of k and ε will be determined later. Let z^* denote the optimal solution value of $MPISP(\pi, c)$ with corresponding solution set S^*. By $S^*(V_i)$ we denote $\{v | v \in S^* \cap v \in V_i\}$. Choosing

$$\ell := \arg \min_{i \in \{1, \ldots, k\}} \left\{ \sum_{v \in S^*(V_i)} p(v) \right\} \tag{11}$$

we get by an averaging argument:

$$\sum_{v \in S^*(V_l)} p(v) \leq \frac{1}{k} z^*$$

Let z_l denote the optimal solution value of $MPISP(\pi, c)$ on the subgraph induced by $\bigcup_{j \neq l} V_j$:

$$(1+\varepsilon) \cdot z^a \geq z_l \geq z^* - \sum_{v \in S^*(V_l)} p(v) \geq z^* - \frac{1}{k} z^* = \left(\frac{k-1}{k} \right) z^*$$

Note that the set of vertices leading to a profit of $z^* - \sum_{v \in S^*(V_l)} p(v)$ is feasible by π being hereditary. For an $(1+\delta)$-approximation algorithm for $MPISP(\pi, c)$ on H-minor free graphs we need $(1+\delta) \cdot z^a \geq z^*$. But we already have:

$$\left(\frac{k}{k-1} \right) \cdot (1+\varepsilon) \cdot z^a = \left(1 + \frac{1}{k-1} + \frac{k}{k-1} \cdot \varepsilon \right) \cdot z^a \geq z^*$$

So let δ be given. If we choose

$$\delta > \left(\frac{1}{k-1} + \frac{k}{k-1} \cdot \varepsilon\right),$$

which is equivalent to

$$\varepsilon \leq \left(\delta \cdot \frac{k-1}{k} - \frac{1}{k}\right)$$

and $k > \frac{1}{\delta} + 1$ (in order to guarantee $\varepsilon > 0$), we get the desired result.

To obtain a PTAS for $EPISP(\pi, c)$ we have to solve the $(1+\varepsilon)$ approximation algorithm on bounded treewidth graphs k times on the disjoint union of the subgraphs induced by V_i and its complement $\overline{V}_i$. Note that by Theorem 12 the subgraphs induced by V_i and $\overline{V}_i$ both have bounded treewidth and therefore also their disjoint union. S^* denotes the set of edges leading to an optimal solution value z^*. Now

$$\ell = \arg \min_{i \in \{1,\dots,k\}} \left\{ \sum_{(v \in V_i, \bar{v} \in \overline{V}_i) \wedge ((v,\bar{v}) \in S^*)} p((v,\bar{v})) \right\}$$

denotes the index giving the minimal profit of the edges included in S^* between V_l and $\overline{V}_l$. Now, one can proceed in analogy to $MPISP(\pi, c)$ and get the desired result for the edge-valued case. □

Pferschy and Schauer [32] showed an FPTAS for KCG on graphs of bounded treewidth which immediately implies the following result.

Corollary 14. *There exist a PTAS for KCG on H-minor free graphs.*

As a prominent example, it follows from the famous Robertson-Seymour Graph Minors Theorem (cf. Diestel [15, Sec. 12]) that graphs which can be embedded without crossing on a surface of constant genus are H-minor free. Naturally, this includes planar graphs.

Corollary 15. *There exist a PTAS for KCG on planar graphs and - more general - on graphs of bounded genus.*

5. The Knapsack Problem with Forcing Graph *KFG*

From a graph theoretic point of view, KFG is equivalent to finding a vertex cover in a graph such that its total weight is not necessarily minimal, but does not exceed the given threshold c. Among all such vertex covers we want to maximize the total profit.

In this section we will show that all FPTAS results we derived for KCG also hold for KFG. To do so we transform KFG into a minimization knapsack problem with conflict graphs ($MKCG$) such that the optimal solutions of both problems coincide. Our pseudopolynomial time algorithms for KCG can be easily transferred to $MKCG$. Then Theorem 1 implies FPTASs for $MKCG$ and thus also for KFG since both belong to the class of subset selection problems.

Note that the transformation of KFG to $MKCG$ is indeed required (at least for weakly chordal graphs) since the algorithm for KCG on graphs whose potential maximal cliques can be listed in polynomial time (see Section 2) cannot be easily extended to KFG, however it can be extented to solve $MKCG$.

Finally, we will state other approximability results for KFG in Section 5.3.

5.1. Transforming KFG into a Minimization Knapsack Problem with Conflict Graphs

The *minimization knapsack problem* (MKP) is a variant of the standard knapsack problem where the sum of profits should be *minimized* but the total weight should be *at least as large* as the given bound c (cf. [24, Sec. 13.3]).

It is well known that if $S \subseteq V$ is the minimum weight vertex cover of a vertex weighted graph $G = (V, E)$, then the complement $V \setminus S$ is a maximum weight independent set. In a similar way we will show that the optimal value of KFG with forcing graph G is equal to the optimal value of the *minimization knapsack problem with conflict graphs* ($MKCG$) with conflict graph G:

$$(MKCG) \qquad \min \sum_{j=1}^{n} p_j x_j \tag{12}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \geq \sum_{j=1}^{n} w_j - c \tag{13}$$

$$x_i + x_j \leq 1 \quad \forall \ (i,j) \in E \tag{14}$$

$$x_j \in \{0,1\} \quad j = 1, \ldots, n. \tag{15}$$

Theorem 16. *The optimal solution values of KFG and MKCG coincide.*

Proof. Let $P = \sum_j p_j$ and $W = \sum_j w_j$. Let z be the optimal solution value of KFG with solution set S. By definition of KFG, S is a vertex cover in G with profit z and a weight $w \leq c$. This implies that $V \setminus S$ is a feasible independent set in G with profit $P - z$ and weight $W - w$. Clearly $V \setminus S$ is feasible for $MKCG$ and also optimal: Otherwise there exists a feasible solution set S' for $MKCG$ with profit value z' and $z' < (P - z)$. But then $V \setminus S'$ is a vertex cover in G which is also feasible for KFG with solution value $P - z' > P - (P - z)$ contradicting the optimality of z for KFG. The other direction follows analogously. □

Hence, any exact algorithm solving $MKCG$ is also an exact algorithm for KFG. Moreover, since the result of Theorem 1 works for both maximization and minimization problems, any pseudopolynomial algorithm for $MKCG$ immediately implies an FPTAS also for KFG.

MKP with demand bound c can be solved in general by dynamic programming by reaching (cf. Kellerer et al. [24]) where for each possible profit value the solution set with maximum weight is calculated. The optimal solution can then be found as the set with minimal profit value and weight exceeding c. The running time of this dynamic programming approach is $O(nP)$. For chordal graphs and graphs of bounded treewidth this algorithm serves as a basis for more complicated dynamic programming schemes solving $MKCG$: the algorithms presented in Pferschy and Schauer [32] solve KCG by applying dynamic programming on the clique tree of a chordal conflict graph and on the tree-decomposition of a conflict graph of bounded treewidth. All the main ideas used in these algorithms carry over directly to $MKCG$ and by Theorem 16 also to KFG. Also the time and space complexities remain the same (Table 2). Note that for graphs of treewidth k a constant factor of 2^{k+1} is hidden in the O-notation.

$KFG/MKCG$	time	space
bounded treewidth	$O(nP^2)$	$O(n + P\log(n))$
chordal	$O((n+m)P^2)$	$O(\min\{m, n\log(n)\}P + m)$

Table 2: Time and space complexities for KFG on special forcing graphs.

Therefore, we conclude:

Theorem 17. *There exists an FPTAS for KFG (resp. MKCG) on forcing (resp. conflict) graphs of bounded treewidth and on chordal forcing graphs.* □

A necessary condition for the application of Theorem 1 is the existence of a feasible solution for every instance of a subset selection problem (cf. Section 1.1). This might be seen as a catch for deriving an FPTAS since for KFG (and also $MKCG$) it is in general $\mathcal{NP}$-complete to decide the existence of a feasible solution.

However, on graphs of bounded treewidth and chordal graphs we can compute a minimum vertex cover in polynomial time (see e.g. [9]) and thus decide the feasibility of an instance in a preprocessing step. The same preprocessing step can be done for all graph classes whose potential maximal cliques can be listed in polynomial time by applying the algorithm presented in Fomin and Villanger [17] with the extension by Lokshtanov et al. [25] (recall Remark 2 in Section 2).

For KFG with weakly chordal forcing graphs we can proceed in a similar way and adapt the pseudopolynomial algorithm given in Section 2 for KCG to $MKCG$, which immediately leads to an FPTAS for KFG. However, the technical details are quite involved. In the same way, also all FPTASs for KCG stated in Section 3.3 carry over into FTPASs for KFG on the same graph classes.

5.3. KFG on Planar and Perfect Graphs

Since the vertex cover problem is known to be $\mathcal{NP}$-hard also on planar graphs (even with degree at most 3), it remains $\mathcal{NP}$-complete to decide whether a given instance of KFG has a feasible solution and thus no polynomial time approximation algorithm can be given for KFG on planar graphs (under $\mathcal{P}{\neq}\mathcal{NP}$). This should be seen in contrast to KCG, where a PTAS was given for planar graphs in Corollary 14.

An important superclass of weakly chordal graphs are perfect graphs. By using the result of Milanič and Monnot [28] concerning the exact weighted independent set problem ($EWIS$) we will show the following negative result for KFG with perfect graphs as forcing graphs. A completely analogous result for KCG was given in Pferschy and Schauer [32].

Theorem 18. *KFG is strongly $\mathcal{NP}$-hard with perfect graphs as forcing graphs.*

Proof. It was shown in [28] that $EWIS$ is strongly $\mathcal{NP}$-complete on perfect graphs (in fact even for bipartite graphs of degree at most 3). Let I be an instance of $EWIS$ in a graph G with vertex weights w_j, which asks if an independent set with total weight exactly w exists. But this is equivalent to the question whether there exists a vertex cover in G with weight $W - w$. Now consider an instance of KFG on G with item profits and weights equal to w_j and $c = W - w$. Then by solving this instance of KFG one can immediately answer the given instance of $EWIS$. □

Acknowledgments

We would like to thank Martin Milanič for valuable discussions.

The research was funded by the Austrian Science Fund (FWF): P23829

References

[1] A. Berry, J.-P. Bordat, and O. Cogis. Generating all the minimal separators of a graph. In *Graph-Theoretic Concepts in Computer Science*, volume 1665 of *Lecture Notes in Computer Science*, pages 167–172. Springer, 1999.

[2] A. Berry, A. Brandstädt, V. Giakoumakis, and F. Maffray. Efficiently recognizing, decomposing and triangulating hole- and diamond-free graphs. *Discrete Applied Mathematics*, 2014. to appear.

[3] H.L. Bodlaender and A.M.C.A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[4] V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM J. Comput.*, 31(1):212–232, 2001.

[5] V. Bouchitté and I. Todinca. Listing all potential maximal cliques of a graph. *Theoretical Computer Science*, 276(1-2):17–32, 2002.

[6] A. Brandstädt and V. Giakoumakis. Maximum weight independent sets in hole-and co-chair-free graphs. *Information Processing Letters*, 112(3): 67–71, 2012.

[7] A. Brandstädt and V. Giakoumakis. Addendum to: Maximum weight independent sets in hole- and co-chair-free graphs. *Information Processing Letters*, 115(2):345–350, 2015.

[8] A. Brandstädt and C.T. Hoàng. On clique separators, nearly chordal graphs, and the maximum weight stable set problem. *Theoretical Computer Science*, 389(1):295–306, 2007.

[9] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* SIAM, 1999.

[10] A. Brandstädt, V.V. Lozin, and R. Mosca. Independent sets of maximum weight in apple-free graphs. *SIAM Journal on Discrete Mathematics*, 24: 239–254, 2010.

[11] A. Brandstädt, V. Giakoumakis, and F. Maffray. Clique separator decomposition of hole-free and diamond-free graphs and algorithmic consequences. *Discrete Applied Mathematics*, 160:471–478, 2012.

[12] K. Cameron, R. Sritharan, and Y. Tang. Finding a maximum induced matching in weakly chordal graphs. *Discrete Mathematics*, 266(1–3):133–142, 2003.

[13] A. Darmann, U. Pferschy, J. Schauer, and G.J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159:1726–1735, 2011.

[14] E.D. Demaine, M.T. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005.*, pages 637–646, 2005.

[15] R. Diestel. *Graph Theory.* Springer, 4th edition, 2012.

[16] G. Even, M. Halldórsson, L. Kaplan, and D. Ron. Scheduling with conflicts: online and offline algorithms. *Journal of Scheduling*, 12(2):199–224, 2009.

[17] F.V. Fomin and Y. Villanger. Finding induced subgraphs via minimal triangulations. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 5 of *Leibniz International Proceedings in Informatics*, pages 383–394, 2010.

[18] T. Gallai. Transitiv Orientierbare Graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[19] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[20] R. Hayward, C. T. Hoàng, and F. Maffray. Optimizing weakly triangulated graphs. *Graphs and Combinatorics*, 5(1):339–349, 1989.

[21] R.B. Hayward, J.P. Spinrad, and R. Sritharan. Improved algorithms for weakly chordal graphs. *ACM Transactions on Algorithms*, 3(2):14, 2007.

[22] M. Hifi and N. Otmani. An algorithm for the disjunctively constrained knapsack problem. *International Journal of Operational Research*, 13(1): 22–43, 2012.

[23] M. Hifi, S. Saleh, and L. Wu. A fast large neighborhood search for disjunctively constrained knapsack problems. In *Proceedings of 3rd International Symposium on Combinatorial Optimization, ISCO 2014*, volume 8596 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2014.

[24] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[25] D. Lokshtanov, M. Vatshelle, and Y. Villanger. Independent set in P_5-free graphs in polynomial time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 570–581, 2014.

[26] V.V. Lozin and M. Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *Journal of Discrete Algorithms*, 6:595–604, 2008.

[27] R.M. McConnell and J.P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:189–241, 1999.

[28] M. Milanič and J. Monnot. The complexity of the exact weighted independent set problem. In *Combinatorial Optimization - Theoretical Computer Science: Interfaces and Perspectives*, pages 393–432. Wiley-ISTE, 2008.

[29] R.H. Möhring and F.J. Rademacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. In *Proceedings of the Workshop on Algebraic Structures in Operations Research*, volume 95 of *North-Holland Mathematics Studies*, pages 257–355. Elsevier, 1984. published also as *Annals of Discrete Mathematics* **19**.

[30] A. Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 22: 401–415, 2010.

[31] T. Öncan, R. Zhang, and A.P. Punnen. The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research*, 40(4):920–930, 2013.

[32] U. Pferschy and J. Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.

[33] U. Pferschy and J. Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.

[34] K. Pruhs and G.J. Woeginger. Approximation schemes for a class of subset selection problems. *Theoretical Computer Science*, 382(2):151–156, 2007.

[35] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

[36] J. Spinrad and R. Sritharan. Algorithms for weakly triangulated graphs. *Discrete Applied Mathematics*, 59(2):181–191, 1995.

[37] R.E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[38] S.H. Whitesides. A method for solving certain graph recognition and optimization problems, with applications to perfect graphs. *Annals of Discrete Mathematics*, 21:281–297, 1984. published also as North-Holland Mathematics Studies **88**.

[39] T. Yamada, S. Kataoka, and K. Watanabe. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43:2864–2870, 2002.

[40] R. Zhang, S.N. Kabadi, and A.P. Punnen. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191–205, 2011.