Sven O. Krumke   Madhav V. Marathe   Diana Poensgen
S. S. Ravi   Hans-Christoph Wirth

Budgeted Maximum Graph Coverage

ZIB-Report 02-24 (May 2002)

Budgeted Maximum Graph Coverage

Sven Oliver Krumke[2], Madhav V. Marathe[3], Diana Poensgen[2], S. S. Ravi[4], and Hans-Christoph Wirth[1]

[1] University of Würzburg
Department of Computer Science, Am Hubland, 97074 Würzburg, Germany
e-mail: wirth@informatik.uni-wuerzburg.de
[2] Konrad-Zuse-Zentrum für Informantionstechnik, Berlin (ZIB)
Takustraße 7, 14195 Berlin-Dahlem, Germany
e-mail: {krumke,poensgen}@zib.de
[3] Los Alamos National Laboratory
P.O.Box 1663, MS 997, Los Alamos, NM 87545, USA
e-mail: marathe@lanl.gov
[4] University at Albany – SUNY
Department of Computer Science, Albany, NY 12222, USA
e-mail: ravi@cs.albany.edu

Abstract An instance of the *maximum coverage* problem is given by a set of weighted ground elements and a cost weighted family of subsets of the ground element set. The goal is to select a subfamily of total cost of at most that of a given budget maximizing the weight of the covered elements.
We formulate the problem on graphs: In this situation the set of ground elements is specified by the nodes of a graph, while the family of covering sets is restricted to connected subgraphs. We show that on general graphs the problem is polynomial time solvable if restricted to sets of size at most 2, but becomes NP-hard if sets of size 3 are permitted. On trees, we prove polynomial time solvability if each node appears in a fixed number of sets. In contrast, if vertices are allowed to appear an unbounded number of times, the problem is NP-hard even on stars. We finally give polynomial time algorithms for special cases where the subgraphs form paths and the host graph is a line, a cycle or a star.

Keywords: budgeted maximum coverage, approximation algorithm, dynamic programming

1 Introduction, Preliminaries, and Related Work

The *budgeted maximum coverage* problem is defined as follows: An instance specifies a set $X = \{x_1, \dots, x_n\}$ of ground elements with weight function $w\colon X \to \mathbb{R}_0^+$, further a family $F \subseteq 2^X$ of *covering sets* with associated costs $c\colon F \to \mathbb{R}_0^+$. The goal is to select a subfamily $F' \subseteq F$ which does not exceed a given constraint on the total cost and maximizes the weight of the covered elements.

The unit cost variant $c \equiv 1$ of the problem is known as the *maximum coverage* problem (see e. g. [Hoc97a] for a survey). A straightforward reduction from the VERTEX COVER problem shows that the unweighted maximum coverage problem is NP-hard even if each ground element appears in no more than 2 sets.

The problem with general cost function $c \not\equiv 1$ has been investigated by Khuller et al. [KMN99]. The authors give an approximation algorithm with performance $(1-1/e) \approx 0.63$ and show that this is best possible unless $\mathrm{NP} \subseteq \mathrm{DTIME}(N^{O(\log\log N)})$.

There is an alternative definition of the budgeted maximum coverage problem used by Ageev et al. [AS99,AS02]: "Given ground elements I and family $F \subseteq 2^I$ with weights

$w\colon F \to \mathbb{R}_0^+$ and an integer $p \in \mathbb{N}$, find a subset $X \subseteq I$ of the ground elements with $|X| = p$ which maximizes the total weight of the sets from F intersecting X." Comparing the two definitions, is appears that the role of ground elements and sets is interchanged. Notice that a set of size k in the Ageev definition transforms into a ground element appearing in k sets in the definition employed by Khuller et al. We will stick to the notation used by Khuller et al. [KMN99] throughout the paper.

Definition 1 (Budgeted Maximum Graph Cover problem)
An instance of Budgeted Maximum Graph Cover *(*GC *for short) is given by an undirected simple graph $G = (V, E)$ with node weight function $w\colon V \to \mathbb{R}$, a weighted family $F = \{S_1, \ldots, S_{|F|}\}$ of connected induced subgraphs S_i with cost function $c\colon F \to \mathbb{R}_0^+$, and a budget value $B \in \mathbb{N}$. The goal is to find a selection $F' \subseteq F$ of subgraphs of total cost*

$$c(F') := \sum_{S \in F'} c(S) \leq B\,,$$

such that the total weight $w(F')$ covered by the selection, defined by

$$w(F') := \sum_{v \in \bigcup_{S \in F'} S} w(v)\,,$$

is maximized.

We assume without loss of generality that each of the subgraphs does not violate the budget constraint, i. e., $c(S) \leq B$ for all $S \in F$. By $\mathrm{GC}^{\mathrm{unit}}$ we denote the set of instances where all sets have cost 1. For a graph class Γ, we use the notion "Γ-GC" to denote the fact that all graphs of the family F belong to graph class Γ. Further, the notion "GC on Γ" means that the input graph G is restricted to graph class Γ. We denote by GC_k the subset of instances of problem GC, where each set has cardinality at most k.

2 Maximum Graph Coverage on General Graphs

This section considers the case of $\mathrm{GC}^{\mathrm{unit}}$ on general graphs. We first address the case of $\mathrm{GC}_2^{\mathrm{unit}}$ where each set has cardinality at most 2 and give a polynomial time algorithm based on matching techniques. We then show hardness of $\mathrm{GC}_k^{\mathrm{unit}}$ for $k \geq 3$.

2.1 $\mathrm{GC}^{\mathrm{unit}}$ with Sets of Cardinality Bounded by 2

Given an instance of problem $\mathrm{GC}_2^{\mathrm{unit}}$, we claim that we can assume without loss of generality that each set has cardinality exactly 2. This can be verified easily: for each singleton set $S = \{v\} \in F$ we can insert a dummy node v' with zero weight into the graph, add a dummy edge joining v and v' and replace S by $S' := \{v, v'\}$. We thus obtain an equivalent instance of $\mathrm{GC}_2^{\mathrm{unit}}$ whose optimal value equals the optimal value of our original instance. Thus, for the remainder of the section we will assume that we are only given sets of size 2. Due to this observation, each set $S \in F$ corresponds to an edge in G. We will also assume in the sequel that G does not contain "useless edges", that is, edges which are not sets from F. The following lemma shows that the graph cover problem can be reduced to a matching problem with cardinality constraint:

Lemma 2
$\mathrm{GC}_2^{\mathit{unit}}$ *with input graph $G = (V, E)$ and budget constraint B is equivalent to the problem of finding a maximum weight matching in a graph with $2|V|$ vertices and $|E| + |V|$ edges subject to the constraint that the matching contains exactly B edges.*

Proof. Consider the graph H consisting of all vertices and edges in G and, in addition, for each vertex $v \in V$ a new vertex v', called the "mate" of v, which is joined to v by an edge of weight $w(v)$ (hence the only neighbor of a mate note v' is v itself). The weight of an edge (u, v) in H which originates from the edge $(u, v) \in E$ is set to $w(u) + w(v)$.

Let $S = \{S_1, \ldots, S_B\}$ denote an optimal solution for the given instance of GC_2 on graph G, which covers a total weight of W^*. Observe that without loss of generality we can assume that the edge set S decomposes into node disjoint stars, i. e., there is no path of length 3 formed by edges from S.

We construct a matching in H of weight at least W^* which uses exactly B edges. View the sets $\{S_1, \ldots, S_B\}$ as edges in H and denote by H_S the subgraph of H containing all vertices of H and the edges in S. Obviously, if each vertex in H_S has degree at most one, then the edges in S form a matching. In this case, since no two edges in S share a common endpoint, the weight covered by the sets in S equals that of the corresponding matching.

It remains to handle the case where there are vertices of degree greater than 1 in H_S. Let $v \in H$ be such a vertex, u be one of its neighbors in H_S. Then the degree of u equals 1, otherwise there would be a path of length 3. Hence we can replace edge (v, u) by edge (u, u') thus decreasing the degree of v by one. By repeating this replacement procedure, we end up with a matching in graph H. It is easy to observe that the weight of the edges of this matching equals the weight of the nodes covered by S.

Conversely, let H_S be an arbitrary matching in H of total weight W, consisting of B edges. Obviously, the total weight of the nodes from G incident with edges from H_S equals W. To construct a valid covering, replace each edge (v, v') of H_S incident with a mate node v' by an arbitrary edge incident with v. This is possible, otherwise v would be an isolated node in G. Obviously, this replacement operation does not shrink the set of covered nodes from G. □

The following result shows how to solve a maximum weight matching problem with cardinality constraint.

Lemma 3
The maximum weight cardinality k constrained matching problem in a graph $H = (V_H, E_H)$ can be solved in polynomial time by computing a maximum weight perfect matching in an auxiliary graph with $2|V_H| - 2B$ vertices and $|E_H| + |V_H| \cdot (2|V_H| - 2B)$ edges.

Proof. We construct the auxiliary graph H' mentioned in the statement of the lemma as follows: H' contains all vertices and edges of H. Moreover, we add a set X of $|V_H| - 2B$ new vertices, each of which is connected to all vertices in V_H. The weight of the edges between vertices in X and V_H is set to 0, the original edges from E_H retain their weights.

Clearly, a matching of cardinality B can be augmented to a perfect matching in H' of the same weight by adding, for each of the $|V_H| - 2B$ unmatched vertices in V_H an edge to a vertex in X. Conversely, any perfect matching in the auxiliary graph H' must contain exactly $|V_H| - 2B$ edges between vertices in X and V_H. Hence, we can derive from this matching a matching of cardinality exactly k in the original graph H. Since all edges between X and V_H had zero weight, this matching in H has the same weight as the perfect matching in H'. □

From Lemma 2 and Lemma 3 we can immediately conclude:

Theorem 4 (Solving $\mathrm{GC}_2^{\mathrm{unit}}$)
GC_2^{unit} *can be solved in polynomial time.* □

2.2 GC with Sets of Cardinality Bounded by $k \geq 3$

The results of the previous section for $\mathrm{GC}_2^{\mathrm{unit}}$ are now complemented by a hardness result which shows that the problem gets NP-hard when the bound on the size of the sets increases.

Theorem 5 (Hardness of $\mathrm{GC}_k^{\mathrm{unit}}$)
GC_k^{unit} is NP-hard to solve for any fixed $k \geq 3$, even when restricted to path-GC_k^{unit} on bipartite graphs.

Proof. We use a reduction from 3-Dimensional Matching which is well known to be NP-complete (see [GJ79, Problem SP1]). The reduction shows the hardness of Path-GC_3 on bipartite graphs and easily extends to any fixed $k \geq 3$.

An instance of 3-Dimensional Matching is given by a set $M \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number q of elements. The question posed is, whether M contains a matching, i.e., a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate.

We construct the natural bipartite graph G with vertex set $W \cup X \cup Y$ and edge set $W \times X \cup X \times Y$. For any triple $(w, x, y) \in M$ the collection F contains a the set $S = \{w, x, y\}$ which forms a path in G. The budget is set to $B := q$. It is easy to see that M contains a matching if and only if there is a cover in G with q sets from F which covers the whole graph. □

3 Maximum Graph Coverage on Paths and Cycles

In view of the results of Section 2 we consider restricted families of host graphs. We start with the imaginably simplest case, namely the restriction of the input graph to being a path. Obviously, also the family of subgraphs consists of paths in this case. It will turn out that even this very simple setting leads to an NP-hard optimization problem (Theorem 14), while the unit cost variant of the problem is solvable in polynomial time (Theorem 8).

3.1 Unit Cost Function

We first consider the situation where the cost of each set is one. We introduce some easy observations. Let $G = (V, E)$ be an instance of $\mathrm{GC}^{\mathrm{unit}}$ on paths with family $F \subseteq 2^V$ of subgraphs. Let $V = \{v_1, \ldots, v_n\}$ be the set of nodes and $E = \{\, (v_i, v_{i+1}) \mid i = 1, \ldots, n-1 \,\}$ be the set of edges of the input graph. For an arbitrary subgraph $S \in F$, denote by

$$l(S) := \min\{\, i \mid v_i \in S \,\}$$

the node of subgraph S with the lowest index. For arbitrary subgraphs $S_i, S_j \in S$, we say that S_j *dominates* S_i if $S_j \supset S_i$ and $l(S_i) = l(S_j)$. It is easy to see that the following assumptions can be made without loss of generality:

Assumption 6 (Valid for $\mathrm{GC}^{\mathrm{unit}}$ on paths)
1. *Family F contains no dominated sets.*
2. *The size of the family is bounded by $|F| \leq n$.*
3. *The budget is bounded by $B \leq |F|$.*
4. *For $F = \{S_1, \ldots, S_{|F|}\}$ we have $l(S_1) < \cdots < l(S_{|F|})$.*
5. *We have $v_1 \in S_1$ and $v_n \in S_{|F|}$.*

We will now develop a dynamic programming scheme for solving the problem. Define

$$f(b,i) \qquad (\text{where } 0 \le b \le B \text{ and } 1 \le i \le n)$$

to be the maximum weight of nodes from the restricted node set $\{v_1,\dots,v_i\}$ which can be covered by up to b subgraphs from F. Let $F'(b,i) \subseteq F$ be a corresponding selection implementing that value. By $S(b,i)$ we denote the set S of maximal index from $F'(b,i)$ which covers v_i; if v_i is not covered by $F'(b,i)$ at all then we set $S(b,i) := \emptyset$. From this definition it follows that $f(B,n)$ is the value of an optimal solution.

It is easy to observe that for $i = 1$ we have

$$f(b,1) = \begin{cases} w(v_1), & \text{if } b > 0, \\ 0, & \text{otherwise.} \end{cases} \qquad \text{and} \qquad S(b,1) = \begin{cases} S_1, & \text{if } b > 0, \\ \emptyset, & \text{otherwise.} \end{cases}$$

We now define for $i = 1,\dots,n-1$

$$f(b,i+1) := \begin{cases} f(b,i) + w(v_{i+1}), & \text{if } v_{i+1} \in S(b,i), \\ \max\{f^*(b,i), f(b,i)\}, & \text{otherwise,} \end{cases} \tag{1}$$

where we use the shortcut

$$f^*(b,i) := \max_{\substack{S \in F \\ v_{i+1} \in S}} \big(w(S \cap \{v_1,\dots,v_{i+1}\}) + f(b-1, l(S)-1)\big)\,. \tag{2}$$

We claim now that Definition (1) is compatible with the above definition of $f(.,.)$, i. e., the dynamic program maintains an optimal solution at each stage. To this end, let $\widetilde{\mathrm{OPT}} := \{\tilde{S_1},\dots,\tilde{S_B}\}$ denote an optimal solution to the instance, $\mathrm{OPT} := w(\widetilde{\mathrm{OPT}})$ its value. Assume that $l(\tilde{S_1}) < \cdots < l(\tilde{S_B})$. Restricted to the subgraph induced by nodes $v_1,\dots,v_i$, this solution induces costs

$$\tilde{c}(i) := \begin{cases} \min\{\, b \mid v_i \in \tilde{S}_b \,\}, & \text{if } v_i \text{ is covered by } \widetilde{\mathrm{OPT}}, \\ \tilde{c}(i-1), & \text{otherwise.} \end{cases}$$

(where $\tilde{c}(0) := 0$) and contributes a weight of

$$\tilde{w}(i) := w\Big(\bigcup_{\tilde{S} \in \widetilde{\mathrm{OPT}}} (\tilde{S} \cap \{v_1,\dots,v_i\})\Big)\,.$$

Our claim is justified by the following lemma which can be proved by induction.

Lemma 7
For all $i = 1,\dots,n$, we have $f(\tilde{c}(i),i) \ge \tilde{w}(i)$.

Proof. Let $\widetilde{\mathrm{OPT}}$ be an optimal solution. We prove the claim by induction on i. Case $i = 1$ can be verified by inspection. Assume that the claim is valid for some $i \ge 1$.

First consider the case where $v_{i+1} \in S(\tilde{c}(i+1), i)$. Then we conclude

$$\begin{aligned} f(\tilde{c}(i+1), i+1) &= f(\tilde{c}(i+1), i) + w(v_{i+1}) \\ &\ge f(\tilde{c}(i), i) + w(v_{i+1}) && \text{by monotonicity of } f(.,i) \\ &\ge \tilde{w}(i) + w(v_{i+1}) \ge \tilde{w}(i+1) && \text{by induction hypothesis}\,. \end{aligned}$$

The remaining case, $v_{i+1} \notin S(\tilde{c}(i+1), i)$, splits in two sub-cases. On the one hand, assume that v_{i+1} is covered by the optimal solution. Let $S \in \widetilde{\mathrm{OPT}}$ be the set with $v_{i+1} \in S$. Then the budget spent by the optimal solution satisfies

$$\tilde{c}(i+1) = \tilde{c}(l(S)-1) + 1\,, \tag{3}$$

and the total weight gained can be distributed in the same manner. Hence,

$$\begin{aligned} f(\tilde{c}(i+1), i+1) &\geq w(S) + f(\tilde{c}(i+1)-1, l(S)-1) && \text{by the above observation} \\ &= w(S) + f(\tilde{c}(l(S)-1), l(S)-1) && \text{by (3)} \\ &\geq w(S) + \tilde{w}(l(S)-1) \geq \tilde{w}(i+1) && \text{by induction hypothesis} \end{aligned}$$

On the other hand, node v_{i+1} is not covered by $\widetilde{\mathrm{OPT}}$. Then we have $\tilde{c}(i+1) = \tilde{c}(i)$ and $\tilde{w}(i+1) = \tilde{w}(i)$ and the claim follows immediately. □

Theorem 8 (Solving GC$^{\mathbf{unit}}$ on paths)
GC^{unit} *on paths can be solved in polynomial time.*

Proof. The claim of the theorem immediately follows from Lemma 7 for the case $i = n$, using the observation that $\tilde{c}(n) = B$ and $\tilde{w}(n) = \mathrm{OPT}$. □

This result can be generalized. The algorithm behind Theorem 8 can be used to solve the problem on cycles. There are n ways to break up a cycle of n nodes into a path. Each of these n instances can be solved using the above algorithm. At the end, we take the best solution as the solution on the original cycle.

Corollary 9 (Solving GC$^{\mathbf{unit}}$ on cycles)
GC^{unit} *on cycles can be solved in polynomial time.* □

3.2 General Cost Functions

If the constraint $c \equiv 1$ on the cost function is relaxed, the problem becomes NP-hard (Theorem 14). However, we will show that there is a PTAS. To this end we use a commonly used technique of employing a (pseudopolynomial) algorithm for the dual problem together with a binary search to construct an algorithm for the original problem. Scaling the input parameters then yields a polynomial time approximation algorithm.

Solving the Dual Problem The dual problem is characterized as follows: Given a positive number $W \in \mathbb{N}$, find a subfamily $F' \subseteq F$ of total weight $w(F') \geq W$, while the budget $c(F')$ needed by the solution is minimized.

We solve this problem by dynamic programming: Define

$$g(w, i) \qquad (\text{where } 0 \leq w \leq W \text{ and } 1 \leq i \leq n)$$

to be the minimal budget which must be spent on the subgraph with nodes $v_1, \ldots, v_i$, such that the covered weight on this subgraph is at least w. For $i = 1$ we observe that

$$g(w, 1) = \begin{cases} c(S_1), & \text{if } w(v_1) \geq w, \\ \infty, & \text{otherwise.} \end{cases}$$

To calculate $g(w, i+1)$ one can observe that the previous solution implementing the value of $g(w, i)$ is also valid for the new situation. However, there might be a solution with a strictly lower budget: This solution must cover the new node v_{i+1} by a subset $S \in F$ since otherwise the solution would have been discovered in the step before. The cost of such a solution is distributed to $c(S)$ and the remaining graph consisting of the nodes $v_1, \ldots, v_{l(S)-1}$. Hence we define

$$g^*(w,i) := \min_{\substack{S \in F \\ v_{i+1} \in S}} c(S) + g(w - w(S \cap \{v_1, \ldots, v_{i+1}\}), l(S) - 1)$$

and claim that

$$g(w, i+1) = \min\{g^*(w,i), g(w,i)\} \qquad \text{for all } 0 \le w \le W\,. \tag{4}$$

Thus, $g(W,n)$ is the cost of the optimal solution. It is easy to see that the running time of the dynamic program is in $O(Wn^2)$, i. e., the algorithm for the dual problem has pseudopolynomial running time.

Lemma 10
Equation (4) is valid, i. e., the dynamic program approach is correct.

Proof. Omitted in this abstract. □

Solving the Primal Problem We now come back to the original problem which is given a bound B on the available budget. Observe that no valid solution can cover more weight than a total of $W_{\text{tot}} := \sum_{v \in V} w(v)$. Hence, we use a binary search to find the largest $W \in [1, W_{\text{tot}}]$ such that the algorithm from above, given W as the weight constraint, finds a solution which satisfies the budget constraint. The solution found by this procedure is an optimal solution of the original problem. The running time is in $O(\log W_{\text{tot}} \cdot W_{\text{tot}} \cdot n^2)$.

Corollary 11
Problem GC *on paths can be solved in pseudopolynomial time.* □

Approximating the Problem At this point one can apply a well known scaling technique to derive a fully polynomial time approximation scheme (FPAS) for GC on paths. Let $\varepsilon > 0$ be an arbitrary accuracy parameter. Define scaling parameter M by

$$M := \frac{W_{\max}}{n(1 + 1/\varepsilon)} \qquad \text{where } W_{\max} := \max_{v \in V} w(v)\,.$$

Given instance I with weight function $w\colon V \to \mathbb{R}^+$, setup a scaled instance I' with weight function w' by defining

$$w'(v) := \lfloor w(v)/M \rfloor \qquad \text{for all } v \in V\,.$$

Notice that

$$w(v)/M - 1 \le w'(v) = \lfloor w(v)/M \rfloor \le w(v)/M\,. \tag{5}$$

If we solve the problem on the scaled instance I', we get an optimal solution of weight OPT$'$. This is performed in polynomial running time, which follows from the fact that the total weight on the scaled instance is bounded by

$$W'_{\text{tot}} = \sum_{v \in V} w'(v) \le \frac{\sum_{v \in V} w(v)}{W_{\max}} n(1 + 1/\varepsilon) \le n^2(1 + 1/\varepsilon)\,.$$

Now we interpret the solution found on the scaled instance as a solution of the original instance. Let W be the weight of this solution under the unscaled cost function w.

Lemma 12
The inequality $\frac{OPT}{W} \leq \frac{W+Mn}{W} \leq 1+\varepsilon$ *holds true.*

Proof. Denote by OPT the weight of an optimal solution, and let $O \subseteq V$ be the subset of nodes covered. From Equation (5) we conclude

$$W \geq M \cdot \mathrm{OPT}' \tag{6}$$

and, similarly,

$$\mathrm{OPT} = \sum_{v\in O} w(v) \leq \sum_{v \in O}(w'(v)+1)M \leq M \sum_{v\in O} w'(v) + Mn \leq M \cdot \mathrm{OPT}' + M \cdot n, \tag{7}$$

where the sum is taken over all nodes which are covered by the solution.

Now observe that the weight covered by an optimal solution is bounded from below by

$$\mathrm{OPT} \geq W_{\max}\,, \tag{8}$$

since otherwise one could replace the current optimal solution by a strictly better solution containing only one set $S \in F$ which covers a node of weight $W_{\max}$.

We are now enabled to estimate the approximation performance of the approach:

$$\begin{aligned} \frac{\mathrm{OPT}}{W} &\leq \frac{W+Mn}{W} = 1 + \frac{Mn}{W} \leq 1 + \frac{Mn}{\mathrm{OPT}-Mn} && \text{by (6) and (7)} \\ &\leq 1 + \frac{Mn}{W_{\max}-Mn} = \frac{W_{\max}}{W_{\max}-Mn} && \text{by (8)} \\ &= \frac{W_{\max}}{W_{\max} - \frac{W_{\max}}{n(1+1/\varepsilon)}n} = \frac{1+1/\varepsilon}{1+1/\varepsilon-1} = 1+\varepsilon \end{aligned}$$
□

The previous lemma now implies the main result of this section:

Theorem 13 (Approximating general cost GC on paths)
There is a FPAS for GC *on paths.* □

Hardness of the Problem A straightforward reduction from Knapsack (see [GJ79, Problem MP9]) shows the hardness of GC on paths:

Theorem 14 (Hardness of GC with general cost function)
GC *is NP-hard to solve, even when restricted to* GC_1 *on paths.* □

4 Maximum Graph Coverage on Trees

The following result shows that GC on trees is intractable as long as the cover sets of the instance are allowed to be (very simple shaped) trees themselves.

Theorem 15 (Hardness of GC on trees)
GC^{unit} *is NP-hard even on a star and even if all vertices have weight one.*

Proof. The claim follows from a straightforward reduction from Exact Cover by 3-Sets (X3C, see [GJ79, Problem SP2]). Use the set X of ground elements of the X3C instance as the node set of the graph, and augment the node set by a new center node. Construct a star by connecting each node by an edge to the center. Each covering set defines in the star graph a sub-star with 3 rays in an obvious manner. The budget is set to $B := |X|/3$. □

Due to this result, we will restrict the investigation to path shaped covering sets in the following section.

4.1 Path-GC$^{\text{unit}}$ on Stars

In this section we consider problem path-GC$^{\text{unit}}$ on stars, which can be equivalently formulated as problem path-GC$_3^{\text{unit}}$ on stars. We derive a polynomial time algorithm based on the algorithm for GC$_2^{\text{unit}}$ given in Section 2.

Lemma 16
*On stars, any instance of path-*GC$^{\text{unit}}$ *can be solved by solving at most* $\lfloor |F|/2 + 1 \rfloor$ *instances of* GC$_2^{\text{unit}}$, *where* $|F|$ *is the number of cover sets.*

Proof. Let $(G = (V, E), F)$ be the input graph and covering family. Denote by v_0 the center node of the star shaped graph G. First assume that F contains no singleton sets. In other words, each set $S \in F$ satisfies $|S| \geq 2$. Hence, each set contains the center node v_0. Moreover, we can assume that each set has cardinality 3: this can be achieved by adding a dummy ray with a dummy endpoint of zero weight to the star and augmenting all sets of cardinality 2 by that dummy node.

Since node v_0 is contained in each set and thus covered anyway, we can remove v_0 from each set of the family F. This yields an instance of GC$_{=2}$ which is solvable in polynomial time according to Theorem 4. The solution for (G, F) is obtained afterwards by adding the center node and incrementing the total weight by $w(v_0)$.

We now handle the case where family F contains singleton sets. Then, any optimal solution either consists solely of singleton sets (which can be chosen by a simple greedy algorithm) or the center node v_0 is covered by at least one non-singleton set. If we know in advance that some non-singleton set S_0 is part of the optimal solution, we can reset the weight of the nodes from set S_0 to zero, decrease budget B by 1, augment all sets including the singletons to sets of cardinality 3 and solve the remaining instance as described above. In order to determine such a set S_0, we can perform the test for all of the at most $|F| - 1$ non-singleton sets and take the best solution.

The following observation shows how to reduce the number of tests. Assume that $S_0 \in F$ is a non-singleton set of maximal weight, i. e., $w(S_0) \geq w(S)$ for all $S \in F$. Let $S_0 = \{v_0, x_0, y_0\}$. Let $F^* \subseteq F$ be any solution. We claim that we can transform F^* into a solution F' which covers both nodes x_0 and y_0 and satisfies $w(F') \geq w(F^*)$: Assume that node x_0 is not covered by F^*. If node y_0 is covered by some set $S_1 \in F^*$, then $w(S_1) \leq w(S_0)$, and replacing S_1 by S_0 yields solution F' without decreasing the total weight. Otherwise, the same argument holds even for arbitrary chosen set S_1. This proves the claim.

To this end, let κ be the number of non-singleton sets of maximal weight, let $S_1, \ldots, S_\kappa$ be the collection of that sets, where $S_i = \{v_0, x_{2i-1}, x_{2i}\}$. Denote by $n(x_i)$ the number of sets from family F which contain node x_i. The number of sets from F which share all of its points with $\{v_0, x_1, \ldots, x_{2\kappa}\}$ is bounded from above by $\min\{2\kappa^2, F\}$. An averaging argument shows that there is a foot $x^* \in \{x_1, \ldots, x_{2\kappa}\}$ which is covered by no more than

$$\frac{1}{2\kappa}\sum_{i=1}^{2\kappa} n(x_i) \leq \frac{1}{2\kappa}(|F| + \min\{2\kappa^2, |F|\}) \leq \min\left(\frac{|F|}{2\kappa} + k, \frac{|F|}{k}\right)$$

sets from F. Observe that this number attains its maximum value $|F|/2 + 1$ for $k = 1$. Since we can assume that foot x^* is contained in an optimal solution, it suffices to perform the test on at most $\lfloor |F|/2 + 1 \rfloor$ instances as described above. □

Corollary 17
*Path-*GC$^{\text{unit}}$ *on stars is polynomially solvable.* □

4.2 GC$^{\text{unit}}$ on Trees with Elements of Bounded Frequency

Recall Theorem 15 which shows that GC$^{\text{unit}}$ on trees is hard to solve. Observe that the instance used in that reduction contains elements which appear in many (or all) covering sets. We show that bounding the frequency of elements can be exploited to attack the problem.

Let $I = (G, F)$ be an instance of GC, where $G = (V, E)$ is a tree. For arbitrary node $v \in V$, denote by $\phi_v := |\{\, S \in F : v \in S \,\}|$ the *frequency* of v, i. e., the number of covering sets containing v. By $\phi(F) := \max_{v \in V} \phi_v$ we denote the *maximum frequency* of the family F. The remainder of this section is devoted to proving the following result:

Theorem 18 (Solving GC$^{\text{unit}}$ on trees)
For any fixed $b \in \mathbb{N}$, problem tree-GC$^{\text{unit}}$ restricted to instances with bounded frequency $\phi(F) \le b$ and polynomially bounded weight function w (i. e., $w(x) \in O(poly|X|)$ for all ground elements $x \in X$) can be solved in polynomial time.

We recall the notion of a tree-decomposition (see e.g. [Bod92]):

Definition 19
A tree-decomposition *of a graph $G = (V, E)$ is a pair $D = (\mathcal{S}, T)$ where $\mathcal{S} = \{\, X_i : i \in I \,\}$ is a collection of subsets of V and T is a tree with node set isomorphic to $\mathcal{S}$, such that the following three conditions are satisfied:*

(i) $\bigcup_{i \in V(T)} X_i = V$,
(ii) for all edges $(u, v) \in E$, there exists a subset $X_i \in \mathcal{S}$ containing both vertices u and v,
(iii) for each vertex $v \in V$, the set of nodes $\{\, i : v \in X_i \,\}$ forms a subtree of T.

The width *of the tree-decomposition D is defined to be $\max_{i \in I} |X_i| - 1$. The* treewidth $\mathrm{tw}(G)$ *of a graph G is the minimum width of a tree-decomposition of G.*

In the following, we describe GC$^{\text{unit}}$ on trees as an *integer linear program.* Then we define the *interaction graph* on that program, which describes the correspondence between variables, and we use a result from [HM$^+$02] to show that this interaction graph is of bounded treewidth. This allows us to apply a result from [SH96] providing a polynomial time algorithm for (a class of) integer linear programs with bounded treewidth interaction graphs.

To this end, let Z be a set of variables and $\mathcal{C}$ be a set of constraints on Z. The bipartite graph $\mathrm{BP}(Z, \mathcal{C})$ associated with $(Z, \mathcal{C})$ is the bipartite graph with color classes Z and $\mathcal{C}$, where $z \in Z$ is adjacent to $C \in \mathcal{C}$ if and only if variable z appears in constraint C. The *interaction graph* $\mathrm{IG}(Z, \mathcal{C})$ for $(Z, \mathcal{C})$ is defined to be the graph with vertex set Z, where two vertices are adjacent if and only if they have a common neighbor in the constraint graph.

We consider the following integer linear program formulation of GC$^{\text{unit}}$:

$$\begin{array}{llll}
\text{(IP)} \quad \text{maximize} & \displaystyle\sum_{v \in V} w(v) \cdot y_v & & \\
\text{subject to} & y_v \le \displaystyle\sum_{S \in F : v \in S} x_S & \text{for all } v \in V & (9) \\
& \displaystyle\sum_{S \in F} x_S \le B & & (10) \\
& y_v \in \{0, 1\} & \text{for all } v \in V & \\
& x_S \in \{0, 1\} & \text{for all } S \in F &
\end{array}$$

The following two theorems show that in order to prove that program (IP) is solvable in polynomial time, it suffices to show that it can be re-formulated in an equivalent form which has an associated bipartite graph of bounded treewidth.

Theorem 20 ([SH96])
Let p be a polynomial. Let Z be a set of variables taking values from the domain $\{0,\dots,K\}$, where $K \in \mathcal{O}(p(|Z|))$, and let $\mathcal{C}$ be a set of constraints on Z. Then, for any fixed $k \in \mathbb{N}$ and any nonnegative vector $c = (c_z)_{z\in Z} \in \{0,\dots,p(|Z|)\}^Z$, the integer program of maximizing $\sum_{z\in Z} c_z \cdot z$ subject to the constraints $\mathcal{C}$, restricted to those instances where the interaction graph for $(Z,\mathcal{C})$ has bounded treewidth at most k, can be solved in time $K^{\mathcal{O}(k)}$. □

Theorem 21 ([HM$^+$02])
Let Z be a set of variables and $\mathcal{C}$ be a set of constraints on Z. Suppose that each constraint contains at most k variables. Let $BP(Z,\mathcal{C})$ be the bipartite graph associated with $(Z,\mathcal{C})$ and $IG(Z,\mathcal{C})$ be the interaction graph. Then,

$$\operatorname{tw}(IG(Z,\mathcal{C})) \in \mathcal{O}(k \cdot \operatorname{tw}(BP(Z,\mathcal{C})))$$

□

As a first step, we consider the bipartite graph of (IP) without the single constraint node introduced by the budget constraint (10).

Lemma 22
The bipartite graph of (IP) with node set introduced by variables x and y and constraints (9) has a treewidth at most $\phi(F)+1$.

Proof. We construct a tree decomposition $(\mathcal{S},T)$ of width at most $\phi(F)+1$ as follows. The tree T in the decomposition is a copy of the tree G in the given instance $I=(G,F)$ of GC$^{\text{unit}}$ on trees. For each vertex $v \in T$ we define the set $X_v := \{y_v, x_{S_1}, \dots, x_{S_{\phi_v}}, C_v\}$, where $S_1,\dots,S_{\phi_v}$ are the subtrees in F containing v and C_v is the constraint from (9) for vertex v. Clearly, the maximum cardinality of a set X_v is at most $\phi(F)+2$.

We now argue that $(\mathcal{S},T)$ is in fact a tree-decomposition. It is obvious that conditions (i) and (ii) are satisfied. Also, condition (iii) clearly holds for the y_v and C_v vertices, each of which appear in exactly one set. To see that condition (iii) also holds for the x_S-vertices in the bipartite graph, observe that x_S appears exactly in those sets X_v with $v \in S$, that is, on the sets attached to those vertices v which form the subtree S. □

The previous construction shows that if we restrict GC$^{\text{unit}}$ on trees to those instances where $\phi(F)$ is fixed, the constraints (9) will result in a constraint graph with bounded treewidth. However, the budget constraint (10) in its original formulation has $|F|$ variables and a straightforward use will cause the treewidth to become unbounded. Our goal is now to replace constraint (10) by an equivalent set of new constraints and variables such that the bounded treewidth will be preserved.

To this end, we first transform the tree-decomposition of Lemma 22 into a new tree-decomposition where the tree is binary. This can be done without increasing the width of the decomposition (see e.g. [Bod92]). Moreover, we also modify the decomposition in such a way that for each variable x_S there is a leaf containing only x_S. This can also be accomplished without increasing the treewidth. Let $(\mathcal{S}',T')$ be the resulting new tree-decomposition. We root the tree T' at an arbitrary node r. The *depth* of a node w in T' is defined to be the number of edges on the unique path from r to w. For two vertices u and w in T' their *lowest common ancestor* $\operatorname{lca}(u,w)$ is defined to be the node of largest depth which is both an ancestor of u and w in T'.

We now define a set A of new variables and a set $\mathcal{C}$ of new constraints guided by the tree T' as follows. Initially, both A and $\mathcal{C}$ are empty. We maintain a set L of pairs $(\alpha, \text{vertex}(\alpha))$ where α is a variable and $\text{vertex}(\alpha)$ is a vertex in T'. Initially, $L := \{\, (x_S, \text{leaf}(x_S)) : S \in F \,\}$, where $\text{leaf}(x_S)$ is an arbitrary leaf in T' containing solely variable x_S.

As long as $|L| > 1$, we choose two variables α and β from L with the property that the lowest common ancestor w of $\text{vertex}(\alpha)$ and $\text{vertex}(\beta)$ has maximum depth among all possible combinations of variables in L. Let γ be a new variable. We add γ to A. We remove $(\alpha, \text{vertex}(\alpha))$ and $(\beta, \text{vertex}(\beta))$ from L, and add (γ, w) to L. We add the constraint $C_{\alpha,\beta,\gamma}\colon \alpha + \beta = \gamma$ to $\mathcal{C}$.

Let P be the unique path between $\text{vertex}(\alpha)$ and $\text{vertex}(\beta)$ in T' (which passes through their least common ancestor w). For all nodes $u \in P$ we add $\alpha, \beta, \gamma, C_{\alpha,\beta,\gamma}$ to the sets X_u from the tree decomposition. Then the procedure is iterated.

If $L = \{(\delta, \text{vertex}(\delta)\}$ contains only a single element, we finally add the constraint $C_\delta\colon \delta \leq B$ to $\mathcal{C}$. We also add C_δ to $X_{\text{vertex}(\delta)}$.

Let $(\mathcal{S}'', T')$ denote the tree T' together with the modified sets X_u at termination of the above procedure. The following lemma can be proved easily by induction on the number of iterations:

Lemma 23 (Equivalent tree decomposition)
The pair $(\mathcal{S}'', T')$ is a valid tree-decomposition for the bipartite graph associated with the old variables x_S and y_v, the new variables A and the constraints (9) together with the new constraints $\mathcal{C}$.

Moreover, upon termination the set of constraints (9) in conjunction with all constraints $\mathcal{C}$ are equivalent to (9) in conjunction with (10). □

We finally address the width of the decomposition $(\mathcal{S}'', T')$. It is easy to see that each vertex in T' participates in at most two paths where labels are added. Since each addition yields four new labels, this gives us the following result.

Lemma 24 (Bounded treewidth of new constraint graph)
The width of the tree-decomposition $(\mathcal{S}'', T')$ is at most $\phi(F) + 10$. □

Proof (of Theorem 18). By Lemma 23 and Lemma 24, there is a formulation of GC on trees with bounded frequency which has an associated bipartite graph of bounded treewidth. The claim follows then from Theorem 21 and Theorem 20. □

5 Conclusion and Open Problems

Table 1 gives an overview of the variants of the maximum graph coverage problem investigated in this paper. It turns out that the unit-cost problem is easy to solve on paths, while on trees we can expect polynomial time algorithms only for the case of bounded frequency. The most interesting open questions which remain are to settle the complexities of the problems path-GC^{unit} and path-$\text{GC}_k^{\text{unit}}$ for fixed $k \in \mathbb{N}$ (without restrictions on the frequency).

References

[AS99] A. A. Ageev and M. I. Sviridenko · *Approximation algorithms for maximum coverage and max cut with given sizes of parts* · Proceedings of 7th Conference on Integer Programming and Combinatorial Optimization (IPCO'99), Lecture Notes in Computer Science, vol. 1610, 1999, pp. 17–30.

	unit cost			general cost
	on paths	on trees	on general graphs	
path-GC$_1$				NP-hard (even on paths) [*Theorem 14*]
path-GC$_2$		P [*Theorem 4*]	P [*Theorem 4*]	
path-GC$_3$			NP-hard [*Theorem 5*]	
path-GC	P [*Theorem 8*]	?		
tree-GC	(not defined)	P (with restricted frequency) [*Theorem 18*]		
		NP-hard (even on stars) [*Theorem 15*]		

Table1: Overview over the complexity of problem GC.

[AS02] A. A. Ageev and M. I. Sviridenko · *Pipage rounding: a new method of constructing algorithms with proven performance guarantee* · Preliminary version appeared as [AS99], to appear, 2002.

[Bod92] H. L. Bodlaender · *A tourist guide through treewidth* · Tech. Report RUU-CS-92-12, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1992.

[GJ79] M. R. Garey and D. S. Johnson · *Computers and intractability (a guide to the theory of NP-completeness)* · W.H. Freeman and Company, New York, 1979.

[HM$^+$02] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns · *Parallel approximation schemes for a class of planar and near planar combinatorial problems* · Information and Computation (2002).

[Hoc97a] D. Hochbaum · *Approximation covering and packing problems* · in [Hoc97b], 1997.

[Hoc97b] D. S. Hochbaum (ed.) · *Approximation algorithms for NP-hard problems* · PWS Publishing Company, 20 Park Plaza, Boston, MA 02116–4324, 1997.

[KMN99] S. Khuller, A. Moss, and J. Naor · *The budgeted maximum coverage problem* · Information Processing Letters **70** (1999), 39–45.

[SH96] R. E. Stearns and H. B. Hunt III · *An algebraic model for combinatorial problems* · SIAM Journal on Computing **25** (1996), no. 2, 448–476.