Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Fully Proportional Representation with Approval Ballots: Approximating the MaxCover Problem with Bounded Frequencies in FPT Time

Piotr Skowron
University of Warsaw, Poland
p.skowron@mimuw.edu.pl

Piotr Faliszewski
AGH University, Poland
faliszew@agh.edu.pl

Abstract

We consider the problem of winner determination under Chamberlin–Courant's multiwinner voting rule with approval utilities. This problem is equivalent to the well-known NP-complete MaxCover problem (i.e., a version of the SetCover problem where we aim to cover as many elements as possible) and, so, the best polynomial-time approximation algorithm for it has approximation ratio $1 - \frac{1}{e}$. We show exponential-time/FPT approximation algorithms that, on one hand, achieve arbitrarily good approximation ratios and, on the other hand, have running times much better than known exact algorithms. We focus on the cases where the voters have to approve of at most/at least a given number of candidates.

Introduction

We study the complexity of winner determination under Chamberlin–Courant multiwinner voting rule with approval utilities. Chamberlin and Courant (1983) proposed their rule as a mean of electing committees of representatives (e.g., parliaments, university senates, and so on), but this rule has also found many other applications, for example, in building recommendation systems (Lu and Boutilier 2011), as a model of resource allocation (Skowron, Faliszewski, and Slinko 2013b), or as a variant of the facility location problem (see, for example, discussions provided by Procaccia et al. (2008) and Betzler et al. (2013)).

Intuitively, Chamberlin–Courant rule proceeds as follows. Consider a setting where we have some set C of candidates, a collection V of voters, and the goal is to pick a committee of K candidates that, in some sense, best represent the voters. Each voter provides numerical utilities regarding the available candidates. These utilities express how satisfied each voter would be from being represented by each of the candidates (by assumption, a voter can be represented by a single candidate only). Typically, though not always, the voters do not express the utilities directly but rather rank the candidates and the election rule uses a scoring function to compute the utilities from the rankings (e.g., Borda scoring function assigns utility $m - i$ to a candidate that a voter ranks as i'th best among m candidates). Given some set W of K candidates, each voter's representative is the candidate from W to which this voter has assigned the highest utility (if there are several such candidates, the voter is represented by any one of them). Chamberlin–Courant rule picks a set of K candidates that maximizes the sum of the utilities that the voters derive from their representatives.[1]

Computing winners under Chamberlin–Courant rule is NP-hard (see the work of Procaccia et al. (2008) and of Lu and Boutilier (2011)). However, there is a general greedy approximation algorithm that runs in polynomial time and achieves approximation ratio $1 - \frac{1}{e}$ (irrespective of the nature of voters' utility values; see the analysis of Lu and Boutilier (2011)). For the case of utilities derived using Borda scoring fuction, even stronger approximation algorithms exist: Skowron et al. (2013b) have shown a practically useful polynomial-time approximation scheme.

We consider the case of approval utilities, i.e., the case where agents' utilities come from the set $\{0, 1\}$; the agent gives utility 1 to the approved candidates and the utility 0 to the disapproved ones. This case is particularly interesting because providing approval information puts much less burden on the agents than providing preference rankings. Unfortunately, this case is also computationally harder then the variant of the rule with preference rankings and Borda utilities (no polynomial-time approximation algorithm better than the general one is possible unless $\mathrm{P} = \mathrm{NP}$).

Our Contribution. Chamberlin–Courant rule with approval utilties is equivalent to the MaxCover problem. In the MaxCover problem we are given a set N of n elements, a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets of N, and an integer K. The goal is to find a size-at-most-K subcollection of $\mathcal{S}$ that contains (covers) as many elements from N as possible. Now, each voter corresponds to an element in the set N and each candidate corresponds to a set in the family $\mathcal{S}$. If a voter i approves of candidate S_j, we have $i \in S_j$: Picking the set S_j to be included in the MaxCover solution means that the voter i is "covered" (i.e., has a candidate in the committee that he or she approves of). That is, there is a one-to-one mapping between winner-determination under approval-based Chamberlin–Courant rule and the Max-

[1] Chamberlin–Courant rule has some disadvantages as a rule for electing parliaments, but it is still quite an attractive one. For more detailed comparison of multiwinner voting rules we point the reader to the work of Elkind et al. (2014).

Cover problem. In the technical part of the paper we focus on studying the more abstract MaxCover problem, but we explain our various assumptions in terms of the Chamberlin–Courant voting rule.

The standard greedy algorithm for MaxCover that iteratively picks sets that cover most yet-uncovered elements has approximation ratio $1-\frac{1}{e}$ and this is optimal unless $\mathrm{P} = \mathrm{NP}$ (see, e.g., the textbook (Hochbaum 1996) for the algorithm and the work of Feige (1998) for the approximation lower bound). Thus in our work we focus on exponential-time algorithms that, on one hand, achieve arbitrarily good approximation ratios (i.e., are *approximation schemes*) and, on the other hand, have running times significantly better than the known exact algorithms (indeed, we are often interested in fixed-parameter tractable (FPT) approximation schemes, parameterized by the number of sets allowed in the solution)

We consider three variants of the MaxCover problem, depending on the restrictions regarding elements' frequencies (an element frequency is the number of sets it belongs to):

Upper-bounded frequencies. In this variant we assume that there is some constant p such that each element appears in at most p sets. This variant corresponds to winner determination under Chamberlin–Courant rule where each agent approves of at most p candidates (this is quite a natural restriction; often the voters have energy to express approvals for a small set of candidates only and sometimes such upper bounds are even put forward by election rules). For this case we show FPT approximation schemes (deterministic and randomized).

Lower-bounded frequencies. In this variant we require that there is some constant p such that each element belongs to at least p sets. This corresponds to a setting where each voter is required to approve of at least p candidates. For this case we show an improved analysis of the standard, polynomial-time, greedy algorithm.

Unrestricted case. In this variant we put no restrictions on the MaxCover inputs. While we were unable to find FPT approximation schemes in this case (randomized or not), using an approach introduced by Cygan et al. (2009) and Croce and Paschos (2012), we show exponential-time approximation schemes that seamlessly exchange running time for the quality of the approximation.

We omit some of our proofs due to space constraints. All missing proofs are available in the full version of the paper (Skowron and Faliszewski 2013).

Related Work. Winner-determination under Chamberlin–Courant rule received quite a lot of attention in recent years. Its worst-case complexity was studied by Procaccia at al. (2008) (for the case of approval utilities) and by Lu and Boutilier (2011) (for the case of Borda utilities). Lu and Boutilier have also shown the greedy polynomial-time $(1-\frac{1}{e})$-approximation algorithm for the rule. Skowron et al. (2013b; 2013a) have shown a polynomial-time approximation scheme for the case of Borda utilities and evaluated it empirically. Other authors have studied the parameterized complexity of the rule and its complexity in restricted domains (Betzler, Slinko, and Uhlmann 2013; Yu, Chan, and Elkind 2013; Skowron et al. 2013), as well as in online settings (Oren and Lucier 2014).

Chamberlin–Courant rule is, by far, not the only natural way of selecting committees based on approval information. Various such rules are reviewed by Kilgour (2010) and are studied algorithmically, e.g., by LeGrant et al. (2007), Caragiannis et al. (2010), and Aziz et al. (2014). Naturally, there are also many other multiwinner voting rules.

On the technical side, our work provides parameterized complexity analysis of the MaxCover problem. This problem received relatively little attention in the literature, though recently, and independently, Bonnet et al. (2013) also studied its complexity. Their work is similar in spirit to ours, but the only true overlap between the papers is Theorem 2.

On the other hand, researchers often consider MaxVertexCover, a much-restricted variant of MaxCover in which we are given a graph $G = (V, E)$ and an integer K, and we ask for at most K vertices that jointly cover as many edges as possible (i.e., it is a "Max" variant of the standard VertexCover problem). We stress that MaxVertexCover is considerably simpler even than MaxCover with frequencies bounded by two. Currently the best polynomial-time approximation algorithm for MaxVertexCover, due to Ageev and Sviridenko (1999), has approximation ratio of $\frac{3}{4}$. Parameterized complexity of MaxVertexCover was first studied by Guo et al. (2007). The problem was also studied by Cai (2008), who gave the currently best exact algorithm for it, and by Marx, who gave an FPT approximation scheme (2008); interestingly, our more general algorithm is faster than that of Marx (see the discussion after Theorem 1).

Leaving the realm of FPT running time, Croce and Paschos (2012) provide an exponential-time approximation strategy for MaxVertexCover, based on combining exact (exponential-time) algorithms with (polynomial-time) approximation ones. We use a similar idea (also based on the work of Cygan et. al (2009)) for the case of unrestricted MaxCover problem and compare it to their approach.

Preliminaries

In the introduction we have presented a natural one-to-one correspondence between the Chamberlin–Courant rule and the MaxCover problem. We focus on the MaxCover problem as this way our results are useful both to people interested in multiwinner voting and to those interested in the abstract MaxCover problem only. We assume familiarity with standard notions regarding algorithms and (parameterized) complexity theory, but we provide a brief review. For each positive integer n, we write $[n]$ to mean $\{1,\dots,n\}$.

Let $\mathcal{P}$ be an algorithmic problem where, given some instance I, the goal is to find a solution s that maximizes a certain function f. Given an instance I of $\mathcal{P}$, we refer to the value $f(s)$ of an optimal solution s as $\mathrm{OPT}(I)$ (or simply as OPT if the instance I is clear from the context). Let β, $0 < \beta \leq 1$, be some fixed constant. An algorithm $\mathcal{A}$ that given instance I returns a solution s' such that $f(s') \geq \beta\mathrm{OPT}(I)$ is called a β-approximation algorithm for $\mathcal{P}$. Analogously, we define $\mathrm{OPT}(I)$ and the notion of a γ-approximation algorithm, $\gamma > 1$, for the case of a problem $\mathcal{P}'$, where the task is to find a solution that minimizes a given goal function g. Given instance I of some algorithmic problem, we write $|I|$ to denote the length of the standard, efficient encoding of I. In this paper we focus on the following two problems (the former directly models winner-determination under Chamberlin–Courant rule, whereas the latter models a variant of the rule where we measure voter's dissatisfaction; the number of voters that are not represented by someone they approve of).

Definition 1. *An instance $I = (N, \mathcal{S}, K)$ of the MaxCover problem consists of a set N of n elements, a collection $\mathcal{S} = \{S_1, \dots, S_m\}$ of m subsets of N, and nonnegative integer K. The goal is to find a subcollection $\mathcal{C}$ of $\mathcal{S}$ of size at most K that maximizes $\| \bigcup_{S \in \mathcal{C}} S \|$.*

Definition 2. *The MinNonCovered problem is defined in the same way as the MaxCover problem, but the goal is to find a subcollection $\mathcal{C}$ such that $\|N\| - \| \bigcup_{S \in \mathcal{C}} S\|$ is minimal.*

In the decision variant of MaxCover (of MinNonCovered) we are additionally given an integer T (an integer T') and we ask if there is a collection of up to K sets from $\mathcal{S}$ that cover at least T elements (that leave at most T' elements uncovered). MaxVertexCover is a variant of MaxCover where we are given a graph $G = (V, E)$, the edges are the elements to be covered, and vertices define the sets that cover them (a vertex covers all the incident edges). SetCover and VertexCover are special cases of the decision variants of MaxCover and MaxVertexCover, where we have to cover all the elements (all the edges).

MaxCover and MinNonCovered are quite different in terms of approximation. E.g., if there is a solution that covers all the elements, then a β-approximation algorithm for MaxCover can cover a β fraction of them, but a γ-approximation algorithm for MinNonCovered has to cover them all.

Given an instance I of MaxCover (MinNonCovered), we say that an element e has frequency t if it appears in exactly t sets. We mostly focus on the variants of MaxCover and MinNonCovered where there is a given constant p such that each element's frequency is at most p. We refer to these problems as variants with upper-bounded frequencies. (It is tempting to think that MaxCover with frequencies equal to two is simply MaxVertexCover, but in fact it is a considerably richer problem. In the former, two sets can share many elements, while in the latter two vertices may be connected by at most one edge. Thus, MaxCover with frequencies equal to two is closer in spirit to MaxVertexCover on multigraphs.)

Our focus is on (approximation) algorithms that run in FPT time (see the books of Downey and Fellows (1999), Niedermeier (2006), and Flum and Grohe (2006) for details on parameterized complexity theory). Given an instance I of a problem—whose part, say k, is declared as a parameter—an FPT algorithm is required to run in time $f(k)\mathrm{poly}(|I|)$, where f is some computable function and $\mathrm{poly}(\cdot)$ is some polynomial. (For our problems, unless said otherwise, we always take K, the size of the solution, to be the parameter.) There is also a whole hierarchy of hardness classes, $\mathrm{FPT} \subseteq \mathrm{W[1]} \subseteq \mathrm{W[2]} \subseteq \cdots \subseteq \mathrm{W[P]} \subseteq \cdots$. Since our focus is on algorithmic results, we omit the standard (quite involved) definitions of these classes and instead point the reader to the textbook of Niedermeier (2006). Interestingly, VertexCover is well-known to be in FPT, but the decision variant of MaxVertexCover is W[1]-complete (Guo, Niedermeier, and Wernicke 2007).

Worst-Case Complexity Results

To justify seeking FPT approximation algorithms, we first investigate parametrized complexity of the MaxCover problem. For upper-bounded frequencies it is W[1]-hard (because MaxVertexCover is (Guo, Niedermeier, and Wernicke 2007)) and we show that it, indeed, is W[1]-complete. For lower-bounded frequencies it is W[2]-hard and in W[P].

Theorem 1. *(1) For each constant p greater than 2, the MaxCover problem with frequencies upper-bounded by p is* W[1]*-complete (when parameterized by the number of sets in the solution). (2) For each constant p, $p \geq 1$, MaxCover where each element belongs to at least p sets is* W[2]*-hard and belongs to* W[P] *(when parameterized by the number of sets in the solution)*

For parameter T, the number of elements that we should cover, Bläser gave an FPT algorithm (2003) for MaxCover. On the other hand, for parameter $T' = n - T$, i.e., the number of elements we can leave uncovered (this means considering the MinNonCovered problem), we show that the problem is para-NP-complete (i.e., it is NP-complete even for a constant value of the parameter), but becomes W[2]-complete for the joint parameter (K, T').

Theorem 2. *(1) The MaxCover problem is para-*NP*-complete when parameterized by the number T' of elements that can be left uncovered. This holds even if each element's frequency is upper-bounded by some constant p, $p \geq 2$. (2) MaxCover is* W[2]*-complete when parameterized by both the number K of sets that can be used in the solution and the number T' of elements that can be left uncovered.*

Algorithms for the Bounded Frequencies Cases

This section presents our core results, i.e., approximation algorithms for MaxCover/MinNonCovered with bounded frequencies.

The MaxCover Problem with Upper Bounded Frequencies. Our FPT approximation scheme for MaxCover with upper-bounded frequencies works as follows. Given an instance $I = (N, \mathcal{S}, K)$—with frequencies bounded by some constant p—and a required approximation ratio β, the algorithm restricts itself to a number of sets from $\mathcal{S}$ with highest cardinalities, tries all K-element subcollections of these sets, and returns the best one (the size of the restriction depends only on K, p, and β; see Algorithm 1).

Below, we show that this algorithm indeed achieves a required approximation ratio, then we show that our analysis is tight up to the constant factor of $\frac{3}{4}$, and then we compare our algorithm to the FPT approximation scheme for MaxVertexCover of Marx (2008).

Algorithm 1. *Let $(N, \mathcal{S}, K)$ be the input MaxCover instance. Let $\mathcal{A}$ be the set of $\lceil \frac{2pK}{(1-\beta)} + K \rceil$ sets from $\mathcal{S}$ with highest cardinalities. Return a K-element subset of $\mathcal{A}$ that covers most elements.*

Theorem 3. *For each instance $I = (N, \mathcal{S}, K)$ of MaxCover where each element from N appears in at most p sets in $\mathcal{S}$, Algorithm 1 outputs a β-approximate solution in time* $\mathrm{poly}(n,m) \cdot \binom{\frac{2pK}{(1-\beta)}+K}{K}$.

Proof. Establishing the running time is immediate and so we focus on showing the approximation ratio. Consider an input instance I. Let $\mathcal{C}$ be the solution returned by Algorithm 1 and let $\mathcal{C}^*$ be some optimal solution. Let c be an arbitrary function such that for each element e such that $\exists_{S \in \mathcal{C}^*} : e \in S$, $c(e)$ is some $S \in \mathcal{C}^*$ such that $e \in S$. We refer to c as the *coverage function*. Intuitively, the coverage function assigns to each element covered under $\mathcal{C}^*$ (by, possibly, many different sets) a set "responsible" for covering it. We say that S covers e if and only if $c(e) = S$. Let OPT be the number of elements covered by $\mathcal{C}^*$.

We will show that $\mathcal{C}$ covers at least $\beta\mathrm{OPT}$ elements. Naturally, the reason why $\mathcal{C}$ might cover fewer elements than $\mathcal{C}^*$ is that some sets from $\mathcal{C}^*$ may not be present in $\mathcal{A}$, the set of the subsets considered by the algorithm. We will show an iterative procedure that starts with $\mathcal{C}^*$ and, step by step, replaces those members of $\mathcal{C}^*$ that are not present in $\mathcal{A}$ with the sets from $\mathcal{A}$. The idea of the proof is to show that each such replacement decreases the number of covered element by at most a small amount.

Let $\ell = \|\mathcal{C}^* \setminus \mathcal{A}\|$. Our procedure will replace the ℓ sets from $\mathcal{C}^*$ that do not appear in $\mathcal{A}$ with ℓ sets from $\mathcal{A}$. We renumber the sets so that $\mathcal{C}^* \setminus \mathcal{A} = \{S_1, \ldots, S_\ell\}$. We will replace the sets $\{S_1, \ldots, S_\ell\}$ with sets $\{S'_1, \ldots, S'_\ell\}$ defined through the following algorithm. Assume that we have already computed sets $S'_1, \ldots, S'_{i-1}$ (thus for $i = 1$ we have not yet computed anything). We take S'_i to be a set from $\mathcal{A} \setminus (\mathcal{C}^* \cup \{S'_1, \ldots, S'_{i-1}\})$ such that the set $(\mathcal{C}^* \setminus \{S_1, \ldots, S_i\}) \cup \{S'_1, \ldots, S'_i\}$ covers as many elements as possible. During the i'th step of this algorithm, after we replace S_i with S'_i in the set $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$, we modify the coverage function as follows: (1) for each element e such that $c(e) = S_i$, we set $c(e)$ to be undefined; (2)for each element $e \in S'_i$, if $c(e)$ is undefined then we set $c(e) = S'_i$.

After replacing S_i with S'_i, it may be the case that fewer elements are covered by the resulting collection of sets. Let x_i denote the difference between the number of elements covered by $(\mathcal{C}^* \setminus \{S_1, \ldots, S_i\}) \cup \{S'_1, \ldots, S'_i\}$ and by $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$ (or 0, if by a fortunate coincidence there are more elements covered after replacing S_i with S'_i). By the construction of the set $\mathcal{A}$ and the fact that $S_i \notin \mathcal{A}$, each set from $\mathcal{A}$ contains more elements than S_i. We infer that every set from $\mathcal{A} \setminus (\mathcal{C}^* \cup \{S'_1, \ldots, S'_{i-1}\})$ must contain at least x_i elements covered by $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$. Indeed, if some set $S' \in \mathcal{A} \setminus (\mathcal{C}^* \cup \{S'_1, \ldots, S'_{i-1}\})$ contained fewer than x_i elements covered by $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$, S' would have to cover at least $\|S'\| - (x_i - 1) \geq \|S_i\| - (x_i - 1)$ elements uncovered by $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$. But this would mean that after replacing S_i with S', the difference between the number of covered elements would be at most $(x_i - 1)$.

Let $\mathcal{C}_2^*$ denote the set obtained after the above-described ℓ iterations. Since, for each i, the set $(\mathcal{C}^* \setminus \{S_1, \ldots, S_{i-1}\}) \cup \{S'_1, \ldots, S'_{i-1}\}$ is a subset of $\mathcal{C}^* \cup \mathcal{C}_2^*$, we know that, for each i, each set from $\mathcal{A} \setminus (\mathcal{C}^* \cup \{S'_1, \ldots, S'_\ell\})$ (there is $\|\mathcal{A}\| - K$ such sets) must contain at least x_i elements covered by $\mathcal{C}^* \cup \mathcal{C}_2^*$ (there is at most $2\mathrm{OPT}$ such elements). Since each element is contained in at most p sets, we infer that for each i, $x_i(\|\mathcal{A}\| - K) \leq 2\mathrm{OPT}p$ and, as a consequence, $x_i \leq \frac{2\mathrm{OPT}p}{\|\mathcal{A}\|-K} = \frac{2\mathrm{OPT}p(1-\beta)}{2pK}$. Since $\ell \leq K$, we conclude that $\sum_{i=1}^{\ell} x_i \leq 2\mathrm{OPT}pK\frac{(1-\beta)}{2pK} = (1-\beta)\mathrm{OPT}$. That is, after replacing the sets from $\mathcal{C}^*$ that do not appear in $\mathcal{A}$ with sets from $\mathcal{A}$, at most $(1-\beta)\mathrm{OPT}$ elements fewer are covered. This means that there are K sets in $\mathcal{A}$ that together cover at least $\beta\mathrm{OPT}$ elements. Since the algorithm tries all size-K subsets of $\mathcal{A}$, it finds a solution that covers at least $\beta\mathrm{OPT}$ elements. ☐

Proposition 4. *There is a family $\mathcal{I}$ of pairs (I, β) where I is an instance of MaxCover with bounded frequencies and β is a real number, $0 < \beta < 1$, such that for each $(I, \beta) \in \mathcal{I}$, if we use Algorithm 1 to find a β-approximate solution for I, it outputs an at-most $((\frac{3}{4} + \frac{1}{4}\beta)\mathrm{OPT}(I))$-approximate one.*

Let us now restrict the setting to the MaxVertexCover problem and compare our algorithm to that of Marx (2008), which works as follows: If the input graph contains a vertex with a large enough degree then the algorithm outputs K highest-degree vertices; otherwise it solves the problem exactly in FPT time. To achieve approximation ratio β, Marx's algorithm needs time at least $\Omega((\frac{k^3}{1-\beta})^{(\frac{k^3}{1-\beta})})$ and our algorithm needs $\mathrm{poly}(n,m) \cdot \binom{\frac{2pK}{(1-\beta)}+K}{K}$ time. That is, our algorithm is faster and more general (Marx's approach does not generalize; we postpone the exact discussion why this is so until the full version of the paper). On the other hand, the exact part of Marx's algorithm is interesting in its own right.

The MaxCover Problem with Lower-Bounded Frequencies. For MaxCover with lower-bounded frequencies, the standard algorithm (to which we refer as "the greedy algorithm") which in each iteration greedily extends the solution with a set that contains the most yet-uncovered elements, can achieve a better approximation ratio than in the unrestricted case (and our analysis is tight).

Theorem 5. *The greedy algorithm is a polynomial-time $(1 - e^{-\max(\frac{pK}{m},1)})$-approximation algorithm for the MaxCover problem with frequency lower bounded by p, on instances with m elements where we can pick up to K sets.*

Proposition 6. *For each rational α, $\alpha \geq 1$, there is an instance $I(\alpha)$ of MaxCover (with m sets, element frequency lower-bounded by p, K sets to use, and $\frac{pK}{m} = \alpha$) such that on input $I(\alpha)$, The greedy algorithm achieves approximation ratio no better than $(1 - e^{-\frac{pK}{m}})$.*

Theorem 5 has interesting implications. For each α, $0 < \alpha < 1$, let α-MaxCover be a variant of MaxCover where for each instance the ratio $\frac{p}{m}$ is at least α. This problems arises, e.g., if we use approval-based variant of the Chamberlin-Courant rule with a requirement that each voter

Algorithm 2: For MinNonCovered with frequency upper-bounded by p.

Parameters:
$(N, \mathcal{S}, K)$ — input MinNonCovered instance
p — bound on the number of sets each element can belong to
β — the required approximation ratio of the algorithm
ϵ — the allowed probability of achieving worse than β approximation ratio

`Search(`s, *partial*`)`:
  if $s = 0$ **then return** *partial* ;
  $e \leftarrow$ randomly select element not-yet covered by *partial*;
  best $\leftarrow \{\}$;
  foreach $S \in \mathcal{S}$ *such that* $e \in S$ **do**
    sol $\leftarrow$ `Search(`$(s-1)$, *partial* $\cup \{S\}$`)` ;
    if *sol* **is better than** *best* **then** *best* $\leftarrow$ *sol* ;
  return *best*;
`Main()`: run `Search(`$K, \{\}$`)` for $\lceil -\ln\epsilon/(\frac{\beta-1}{\beta})^K \rceil$ times; return the best solution;

must approve at least some constant fraction of the candidates (e.g., 10%). There exists a polynomial-time approximation scheme (PTAS) for this version of the problem.

Theorem 7. *For each α, $0 < \alpha \le 1$, there is a PTAS for α-MaxCover.*

The exact complexity of α-MaxCover is quite interesting. Using the greedy algorithm, we can show that it belongs to the second level of Kintala and Fisher's β-hierarchy of limited nondeterminism (1980). (A problem belongs to the class β^2 if it can be solved using at most $O(\log^2 n)$ nondeterministic bits, where n is the size of the input.) In effect, it is unlikely that α-MaxCover is NP-complete.

Theorem 8. *For each α, $0 < \alpha < 1$, α-MaxCover is in β^2.*

The MinNonCovered Problem with Upper-Bounded Frequencies. Algorithm 2 is a randomized FPT approximation scheme for the task of minimizing the number of uncovered elements (for the case of upper-bounded frequencies). It works as follows: It picks a random uncovered element e and for each of the sets S that contain e, it tries to recursively build a solution that includes S. Since we assume a constant bound on elements' frequencies, this algorithm works in FPT time. Further, if it is possible to cover all the elements, the algorithm finds such a solution (irrespective of the random choices). If not all elements can be covered, it still finds a good solution with high probability.

Theorem 9. *Algorithm 2 outputs a β-approximate solution for the MinNonCovered problem with probability* $(1-\epsilon)$. *The time complexity of the algorithm is* $\mathrm{poly}(n,m) \cdot \lceil -\ln\epsilon/(\frac{\beta-1}{\beta})^K \rceil \cdot p^K$.

Proof. Let $I = (N, \mathcal{S}, K)$ be out input instance of MinNonCovered and fix some β, $\beta > 1$, and ϵ, $0 < \epsilon < 1$. Each element from N appears in at most p sets from $\mathcal{S}$.

By p_s we denote the probability that a single invocation of the function `Search` (from the `Main` function) returns a β-approximate solution. We will first show that p_s is at least $(\frac{\beta-1}{\beta})^K$, and then we will use the standard argument that if we make $\lceil \frac{-\ln\epsilon}{p_s} \rceil$ calls to `Search`, then the best output is a β-approximate solution with probability $(1-\epsilon)$.

Let $\mathcal{C}^*$ be some optimal solution for I, let $N^* \subseteq N$ be the set of elements covered by $\mathcal{C}^*$, and let $U^* = N \setminus N^*$ be the set of the remaining, uncovered elements. Consider a single call to `Search` from the "for" loop within the function `Main`. Let Ev denote the event that during such a call, at the beginning of each recursive call, at least a $\frac{\beta-1}{\beta}$ fraction of the elements not covered by the constructed solution (i.e., the solution denoted *partial* in the algorithm) belongs to N^*. Note that if the complementary event, denoted $\overline{Ev}$, occurs, then `Search` definitely returns a β-approximate solution. Why is this the case? Consider some tree of recursive invocations of `Search`, and some invocation of `Search` within this tree. Let X be the number of elements not covered by *partial* at the beginning of this invocation. If at most $\frac{\beta-1}{\beta}X$ of the not-covered elements belong to N^*, then—of course—the remaining at least $\frac{1}{\beta}X$ of them belong to U^*. In other words, then we have $\frac{1}{\beta}X \le \|U^*\|$ and, equivalently, $X \le \beta\|U^*\|$. This means that *partial* already is a β-approximate solution, and so the solution returned by the current invocation of `Search` will be β-approximate as well. (Naturally, the same applies to the solution returned at the root of the recursion tree.)

Now, consider the following random process $\mathcal{P}$. (Intuitively, $\mathcal{P}$ models a particular branch of the `Search` recursion tree.) We start from the set N' of all the elements, $N' = N$, and in each of the next K steps we execute the following procedure: We randomly select an element e from N' and if e belongs to N^*, we remove from N' all the elements covered by the first[2] set from $\mathcal{C}^*$ that covers e. Let p_{opt} be the probability that a call to `Search` (within `Main`) finds an optimal solution for I, and let $p_{\mathrm{opt}|Ev}$ be the same probability, but under the condition that Ev takes place. It is easy to see that p_{opt} is greater or equal than the probability that in each step $\mathcal{P}$ picks an element from N^*. Let p_{hit} be the probability that in each step $\mathcal{P}$ picks an element from N^*, under the condition that at the beginning of every step more than $\frac{(\beta-1)}{\beta}$ fraction of the elements in N' belong to N^*. Again, it is easy to see that $p_{\mathrm{opt}|Ev} \ge p_{hit}$. Further, it is immediate to see that $p_{hit} \ge (\frac{\beta-1}{\beta})^K$.

Altogether, combining all the above findings, we know that the probability that `RecursiveSearch` returns a β-approximate solution is at most $p_s \ge \mathrm{P}(\overline{Ev}) + \mathrm{P}(Ev)p_{\mathrm{opt}|Ev} \ge p_{\mathrm{opt}|Ev} \ge (\frac{\beta-1}{\beta})^K$. (That is, either the event Ev does not take place and `Search` definitely returns a β-approximate solution, or Ev does occur, and then we lower-bound the probability of finding a β-approximate solution by the probability of finding the optimal one.) To conclude, the probability of finding a β-approximate solution in one of the $x = \lceil -\ln\epsilon/(\frac{\beta-1}{\beta})^K \rceil$ independent invocations of `Search` from `Main` is at least $1 - (1 - (\frac{\beta-1}{\beta})^K)^x \ge 1 - e^{\ln\epsilon} = 1 - \epsilon$. Establishing the running time is clear. ☐

[2] We assume the sets in $\mathcal{C}^*$ are ordered in some arbitrary way.

Algorithm 3: For the MaxCover problem.

Parameters:
$(N, \mathcal{S}, K)$ — input MaxCover instance
X — the parameter of the algorithm
$C = \{\}, C_{best} = \{\}$;
foreach $(K - X)$*-element subset* C *of* $\mathcal{S}$ **do**
  for $i \leftarrow (K - X + 1)$ **to** K **do**
    $Cov \leftarrow \{e \in N : \exists_{S \in C} e \in S\}$;
    $S_{best} \leftarrow \mathrm{argmax}_{S \in \{S_1, \dots, S_m\} \setminus C}$
    $\|\{e \in N \setminus Cov : e \in S\}\|$;
    $C \leftarrow C \cup \{S_{best}\}$
  $C_{best} \leftarrow$ better solution among C_{best} and C;
return C_{best}

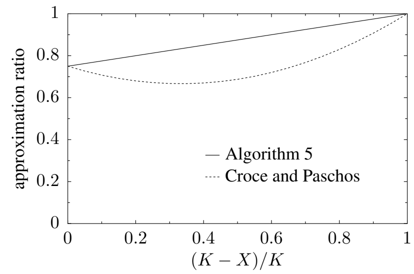

Figure 1: The comparison of the approximation ratios of Algorithm 3 and the algorithm of Croce and Paschos (2012) for MaxVertexCover.

Algorithms for the Unrestricted Case

So far we have focused on the MaxCover problem where element frequencies were either upper- or lower-bounded. Now we consider the unrestricted variant of the problem. We give an exponential-time approximation scheme that, nonetheless, is not FPT. The main idea, which is similar to that of Cygan et. al (2009) and that of Croce and Paschos (2012), is to solve one part of the problem using a brute-force algorithm and to complete the solution using the greedy approximation algorithm (the greedy algorithm for the case of the MaxCover problem; Ageev and Sviridenko's (1999) algorithm if we focus on MaxVertexCover instance).

Theorem 10. *For each MaxCover instance $I = (N, \mathcal{S}, K)$ and each integer X, $0 \leq X \leq K$, Algorithm 3 computes an $(1 - \frac{X}{K}e^{-1})$-approximate solution in time $\binom{m}{K-X} + \mathrm{poly}(K, n, m)$.*

Proof. Let $I = (N, \mathcal{S}, K)$ be our input instance and let $\mathcal{C}^*$, $\mathcal{C}^* \subseteq \mathcal{S}$, be some optimal solution. Let $\mathcal{C}^*_X$ be a subset of $(K - X)$-elements from $\mathcal{C}^*$ that together cover the greatest number of the elements. The sets from $\mathcal{C}^*_X$ cover at least a fraction $\frac{K-X}{K}$ of all the elements covered by $\mathcal{C}^*$. Consider the problem of covering $N \setminus \bigcup_{S \in \mathcal{C}^*_X} S$ with X sets from $(\mathcal{S} \setminus \mathcal{C}^*_X)$. Since $\mathcal{C}^* \setminus \mathcal{C}^*_X$ is an optimal solution for this problem and the greedy algorithm has approximation ratio $1 - \frac{1}{e}$, Algorithm 3's approximation ratio is at least $(\frac{K-X}{K} + \frac{X}{K}(1 - \frac{1}{e})) = (1 - \frac{X}{K}e^{-1})$. □

Corollary 11. *There exists an $(1 - \frac{X}{4K})$-approximation algorithm for MaxVertexCover problem running in time $\binom{m}{K-X} + \mathrm{poly}(K, n, m)$*

It is interesting to compare our Algorithm 3, in the variant tailored to the MaxVertexCover problem, to a similar algorithm of Croce and Paschos (2012). Let $\mathcal{A}_a$ and $\mathcal{A}_e$ be, respectively, an approximation algorithm and an exact algorithm for MaxVertexCover. For a given value X, the algorithm of Croce and Paschos (2012) first uses $\mathcal{A}_e$ to find an optimal solution that uses $K - X$ vertices (out of the K allowed in the solution) and then solves the remaining part of the problem using $\mathcal{A}_a$. If β_a is the approximation ratio of $\mathcal{A}_a$, then this approach gives a $(\frac{X}{K} + \beta_a(1 - \frac{X}{K})^2)$-approximation algorithm. (Our algorithm is very similar, but we run the approximation algorithm for *every possible choice* of $K - X$ initial vertices, and not only for those $K - X$ vertices that cover most edges.)

In Figure 1 we compare the approximation ratios of Algorithm 3 (version from Corollary 11) and of the algorithm of Croce and Paschos. The x-axis represents the parameter $\frac{K-X}{K}$, measuring the fraction of the solution obtained using the exact algorithm (for 0 we use the approximation algorithm alone and for 1 we use the exact algorithm alone). On the y-axis we give approximation ratio of each algorithm. For both algorithms we use the $\frac{3}{4}$-approximation algorithm of Ageev and Sviridenko (1999) and the brute-force algorithm, so for each X, the exponential part of their running times is the same (as far as we know, brute-force search is the fastest polynomial-space exact algorithm for the problem; a faster algorithm, due to Cai (2008), uses exponential space). Under such set-up, Algorithm 3 is superior; trying the approximation algorithm for each initial choice of $K - X$ vertices creates more opportunities to do well.

Conclusions and Future Work

We have studied approximation algorithms for Chamberlin–Courant voting rule, for the case of approval utilties. We have phrased our technical results in terms of the MaxCover problem, but now we take a step back and consider the results from the point of view of multiwinner voting: As long as the elected committee is small (that is, the value K under MaxCover is small) and each voter approves of a small, bounded, number of voters, Chamberlin–Courant rule can be very well approximated (the exponential parts of the running times of our algorithms is low in this setting). If we require that each voter approves of some fraction of the candidates, the standard greedy algorithm becomes a polynomial-time approximation scheme. For the completely unrestricted case, we gave an exponential approximation scheme, in which it is possible to seamlessly exchange the quality of the solution for the running time. Our most interesting open problem is to establish the exact complexity of α-MaxCover.

Acknowledgements. The authors were supported in part by NCN grant DEC-2012/06/M/ST1/00358.

References

Ageev, A., and Sviridenko, M. 1999. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *Proceedings of Integer Programming and Combinatorial Optimization*, 17–30.

Aziz, H.; Gaspers, S.; Gudmundsson, J.; Mackenzie, S.; Mattei, N.; and Walsh, T. 2014. Computational aspects of multi-winner approval voting. In *8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF-14)*.

Betzler, N.; Slinko, A.; and Uhlmann, J. 2013. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research* 47:475–519.

Bläser, M. 2003. Computing small partial coverings. *Inf. Process. Lett.* 85(6):327–331.

Bonnet, E.; Paschos, V.; and Sikora, F. 2013. Multiparameterizations for max k-set cover and related satisfiability problems. Technical Report arXiv:1309.4718 [cs.CC], arXiv.org.

Cai, L. 2008. Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal* 51(1):102–121.

Caragiannis, I.; Kalaitzis, D.; and Markakis, E. 2010. Approximation algorithms and mechanism design for minimax approval voting. In *Proceedings of AAAI-2010*, 737–742. AAAI Press.

Chamberlin, B., and Courant, P. 1983. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review* 77(3):718–733.

Croce, F., and Paschos, V. 2012. Efficient algorithms for the max k-vertex cover problem. *Theoretical Computer Science* 295–309.

Cygan, M.; Kowalik, Ł.; and Wykurz, M. 2009. Exponential-time approximation of weighted set cover. *Inf. Process. Lett.* 109(16):957–961.

Downey, R., and Fellows, M. 1999. *Parameterized Complexity*. Springer-Verlag.

Elkind, E.; Faliszewski, P.; Skowron, P.; and Slinko, A. 2014. Properties of multiwinner voting rules. In *Proceedings of AAMAS-2014*, 53–60.

Feige, U. 1998. A threshold of $\ln n$ for approximating set cover. *J.ACM* 45(4):634–652.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer-Verlag.

Guo, J.; Niedermeier, R.; and Wernicke, S. 2007. Parameterized complexity of vertex cover variants. *Theoretical Computer Science* 41(3):501–520.

Hochbaum, D. 1996. Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems. In Hochbaum, D., ed., *Approximation Algorithms for NP-Hard Problems*. PWS Publishing. 94–143.

Kilgour, D. 2010. Approval balloting for multi-winner elections. In Laslier, J., and Sanver, R., eds., *Handbook on Approval Voting*. Springer. 105–124.

Kintala, C., and Fisher, P. 1980. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal on Computing* 9(1):46–53.

LeGrand, R.; Markakis, E.; and Mehta, A. 2007. Some results on approximating the minimax solution in approval voting. In *Proceedings of AAMAS-2007*, 1193–1195.

Lu, T., and Boutilier, C. 2011. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of IJCAI-2011*, 280–286.

Marx, D. 2008. Parameterized complexity and approximation algorithms. *The Computer Journal* 51(1):60–78.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Oren, J., and Lucier, B. 2014. Online (budgeted) social choice. In *Proceedings of AAAI-2014*, 1456–1462. AAAI Press.

Procaccia, A.; Rosenschein, J.; and Zohar, A. 2008. On the complexity of achieving proportional representation. *Social Choice and Welfare* 30(3):353–362.

Skowron, P., and Faliszewski, P. 2013. Approximating the maxcover problem with bounded frequencies in fpt time. Technical Report arXiv:1309.4405 [cs.DS], arXiv.org.

Skowron, P.; Yu, L.; Faliszewski, P.; and Elkind, E. 2013. The complexity of fully proportional representation for single-crossing electorates. In *Proceedings of the 6th International Symposium on Algorithmic Game Theory*, 1–12.

Skowron, P.; Faliszewski, P.; and Slinko, A. 2013a. Achieving fully proportional representation is easy in practice. In *Proceedings of AAMAS-2013*.

Skowron, P.; Faliszewski, P.; and Slinko, A. 2013b. Fully proportional representation as resource allocation: Approximability results. In *Proceedings of IJCAI-2013*, 353–359.

Yu, L.; Chan, H.; and Elkind, E. 2013. Multiwinner elections under preferences that are single-peaked on a tree. In *Proceedings of IJCAI-2013*, 425–431.