Using Multiwinner Voting to Search for Movies

Grzegorz Gawron
AGH University and VirtusLab
Krakow, Poland
ggawron@virtuslab.com

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

ABSTRACT

We show a prototype of a system that uses multiwinner voting to suggest resources (such as movies) related to a given query set (such as a movie that one enjoys). Depending on the voting rule used, the system can either provide resources very closely related to the query set or a broader spectrum of options. We test our system both on synthetic data and on the real-life collection of movie ratings from the MovieLens dataset.

ACM Reference Format:
Grzegorz Gawron and Piotr Faliszewski. 2021. Using Multiwinner Voting to Search for Movies. In *Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., London, UK, May 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

The idea of multiwinner voting is to provide a committee of candidates based on the preferences of the voters. In principle, such mechanisms have many applications, ranging from choosing parliaments, through selecting finalists of competitions, to suggesting items in Internet stores or services. While the first two types of applications indeed are quite common in practice, the last one, so far, was viewed mostly as a theoretical possibility. Our goal is to change this view. To this end, we design a prototype of a voting-based search system that given a movie (or, a set of movies), finds related ones. The crucial element of our system—enabled by the use of multiwinner voting—is that one may specify to what extent he or she wants to focus on movies very tightly related to the given one, and to what extent he or she wants to explore a broader spectrum of movies that are related in some less obvious ways. Indeed, if someone is looking for movies exactly like the specified one, then using focused search is natural. However, if someone is not really sure what he or she really seeks, or if he or she has already watched the most related movies, looking at a broader spectrum is more desirable.

Viewed more formally, our system belongs to the class of non-personalized recommendation systems based on collaborative filtering. That is, from our point of view the users posing queries are anonymous and we do not target the results toward particular individuals, but rather we try to find movies related to the ones they ask about. In this sense, we provide more of a search-support tool than a classical recommendation system.

To find the relationships between the movies, we use a dataset of movie ratings (in our case, the MovieLens dataset [16]). Such a dataset consists of a set of agents who rate the movies on a scale between one and five stars (where one is the lowest score and five is the highest). For each movie we consider which agents enjoyed it and what other movies these agents liked. More specifically, given the raw data with movie ratings we form a global election where we indicate which users liked which movies (we say that a user liked a movie if he or she gave it at least four stars; in the language of voting literature, liking a movie corresponds to *approving* it). Then, given a query, i.e., either a single movie or a set of movies, we restrict this election to the agents who liked the movies from the query (and the movies they liked, except for the ones from the query). Based on this local election, for each user and each movie that he or she likes, we determine a utility score which indicates how relevant the movie is (briefly put, we need to distinguish between movies that are globally very popular, such as, e.g., *The Lord of the Rings*, from the ones that are mostly popular among the agents in the local election). Finally, we seek a winning committee with respect to one of the OWA-based multiwinner voting rules discussed by Skowron et al. [33], and output its contents as our result (see also the works of Aziz et al. [2] and Bredereck et al. [5]). Since, in general, our rules are NP-hard to compute, we use approximation algorithms and heuristics.

OWA-based rules are parameterized by the *ordered weighted average* operators of Yager [35] and, depending on the choice of these operators, they may provide committees of very similar, individually excellent candidates, or of more diverse ones (see, e.g., the simulations of Elkind et al. [10] and Faliszewski et al. [13], or the theoretical analyses of Aziz et al. [2] and Lackner and Skowron [20**?**]). Thus, by choosing the OWA operators appropriately, we either find movies very closely related to a given query, or those that form a broader spectrum of related movies. Specifically, we use a family of operators parameterized by a value $p \geq 0$, such that for $p = 0$ we get the most focused results, and for larger p's the results become more broad.

Our Contribution. Our main contribution is designing a voting-based search system and testing it in the context of selecting movies. In particular, we show the following results:

(1) Using movie-inspired synthetic data, we show that, indeed, the rules that are meant to choose closely related movies or more broad committees, do so.
(2) Using the MovieLens dataset , we show that the greedy algorithm is, on average, superior to simulated annealing for computing our committees (but simulated annealing also has positive features).

Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., Aziz, Ceppi, Dickerson, Hosseini, Lev, Mattei, McElfresh, Zick (chairs), May 2021, London, UK. © 2021 Copyright held by the owner/author(s).

We also give examples of the answers provided by the system.

Related Work. Regarding multiwinner voting, we point the readers to the overviews of Faliszewski et al. [14] and Lackner and Skowron [21], and to the work of Elkind et al. [11] (who, in particular, suggeted multiwinner voting to select movies for planes' entertainment systems). While multiwinner voting is not yet a mainstream tool in applications, various researchers have used it successfully. Examples include means of ensuring fairness in recommender systems [7] and social media [23], as well as improved techniques for genetic algorithms [12, 28].

For a broad discussion of modern recommendation systems, we point to the handbook of Ricci et al. [30]; see also the work of Sarwar et al. [32] for an early discussion of collaborative filtering methods. As examples of works on movie recommendations, we mention the papers of Ghosh et al. [15] (which describes a movie recommendation system using Black's voting rule with weighted user preferences), Azaria et al. [1] (which focuses on maximizing the revenue of the recommender), Choi et al. [9] (which discusses recommendations based on movie genres), and Phonexay et al. [27] (which adapts some techniques from social networks to recommendation systems). While most of this literature considers the most tightly connected movies, there are works on recommendation systems that focus on diversifying the results (see, e.g., the work of Kim et al. [19]).

So far, we have not found studies whose results we could directly compare to ours (except, of course, for those looking for the most related movies). We continue to seek such related work and means of validating our results.

2 PRELIMINARIES

Let $\mathbb{R}_+$ denote the set of nonnegative real numbers, and for a positive integer i, let $[i]$ denote the set $\{1, \ldots, i\}$.

Utility and Approval Elections. Let $R = \{r_1, \ldots, r_m\}$ be a set of *resources* and let $N = \{1, \ldots, n\}$ be a set of *agents* (in other papers, the resources are often referred to as the *candidates* and the agents are often referred to as the *voters*). Each agent i has a utility function $u_i : R \to \mathbb{R}_+$, which specifies how much he or she appreciates each resource. We assume that the utilities are comparable among the agents and that the utility of zero means that an agent is completely uninterested in a given resource. We do not normalize utility values, so, for example, some agent may be far more excited about the resources than some other one. Committees are sets of resources, typically of a given size k. For a committee $S = \{s_1, \ldots, s_k\}$ and an agent i, by $u_i(S)$ we mean the vector $(u_i(s_1), \ldots, u_i(s_t))$, where the utilities appear in some fixed order over the resources (this order will never be relevant for our discussion). We write $U = (u_1, \ldots, u_n)$ to denote a collection of utility functions, referred to as a *utility profile*. A *utility election* $E = (R, U)$ consists of a set of resources and a utility profile over these resources (a utility profile implicitly specifies the set of agents). An *approval election* is a utility election where each utility is either 1, meaning that an agent approves a resource, or 0, meaning that he or she does not approve it. For approval elections we typically denote the utility profile as $A = (a_1, \ldots, a_n)$ and call it an *approval profile*. For a resource r_t, we write $A(r_t)$ to denote the set of agents that approve it.

OWA Operators. An *ordered weighted average* (OWA) operator is specified by a vector of nonnegative real numbers, such as $\lambda = (\lambda_1, \ldots, \lambda_k)$, and operates as follows. For a vector $x = (x_1, \ldots, x_k) \in \mathbb{R}^k$ and a vector $x' = (x'_1, \ldots, x'_k)$ obtained by sorting x in the nonincreasing order, we have:

$$\lambda(x) = \lambda_1 x'_1 + \lambda_2 x'_2 + \cdots + \lambda_k x'_k.$$

For example, operator $(1, \ldots, 1)$ means summing up the elements of the input vector, whereas operator $(1, 0, \ldots, 0)$ means taking its maximum element. OWA operators were introduced by Yager [35].

(OWA-Based) Multiwinner Voting Rules. A *multiwinner voting rule* is a function f which, given a utility election E and an integer k, returns a family of size-k winning committees. We focus on OWA-based rules.

Consider a utility election $E = (R, U)$, where $R = \{r_1, \ldots, r_m\}$ and $U = (u_1, \ldots, u_n)$, and an OWA operator $\lambda = (\lambda_1, \ldots, \lambda_k)$. Let S be a size-k committee. We define the λ-score of a committee S in election E to be $\lambda\text{-}score_E(S) = \sum_{i=1}^{n} \lambda(u_i(S))$. We say that a multiwinner rule f is OWA-based if there is a family $\Lambda = (\lambda^{(k)})_{k \geq 1}$ of OWA operators, one for each committee size k, such that for each election E and each committee size k, $f(E, k)$ consists exactly of those size-k committees S for which $\lambda^{(k)}\text{-}score_E(S)$ is highest.

HUV Rules. We are particularly interested in the rules that use OWA operators of the followig form, where $p \geq 0$:

$$\lambda^p = (1, 1/2^p, 1/3^p, \ldots),$$

and we refer to them as *p-Harmonic Utility Voting* rules (p-HUV rules). The name stems from the fact that for $p = 1$, their OWA operators sum up to harmonic numbers. Let us consider three special cases:

(1) For a committee size k, the 0-HUV rule chooses k resources with the highest total utility; indeed, its OWA operator is $(1, \ldots, 1)$. Under approval elections, 0-HUV is the classic *Multiwinner Approval Voting* rule (AV).
(2) The 1-HUV rule uses OWA operators $(1, 1/2, 1/3, \ldots)$; for approval elections this is the *Proportional Approval Voting* rule (PAV) of Thiele [34].
(3) Abusing the notation, ∞-HUV is a rule that uses OWA operators $(1, 0, \ldots, 0)$; for approval elections it is the *Chamberlin–Courant* rule (CC); see the works of Chamberlin and Courant [8], Procaccia et al. [29], and Betzler et al. [4].

In the approval voting setting, these three rules correspond to the three main principles of choosing committees. AV chooses *individually excellent* resources, i.e., those that are appreciated by the largest number of agents; PAV chooses committees that, in a certain formal sense, proportionally represent the preferences of the agents [2, 6], and CC (∞-HUV) focuses on *diversity*, i.e., it seeks a committee so that as many agents as possible appreciate at least one item in the committee. For a more detailed description of these principles, see the overview of Faliszewski et al. [14]; for a focus on approval rules, see the survey of Lackner and Skowron [21] and their work on the opposition between AV and CC [22].

We proceed under two premises. The first one is that the 0-HUV, 1-HUV, and ∞-HUV rules extend the principles of individual excellence, proportionality, and diversity to the setting of utility elections

(the visualizations of Elkind et al. [10] support this view). The second one is that for $p > 1$, the rule p-HUV provides committees that achieve various levels of compromise between those of 1-HUV and ∞-HUV (this is supported by the results of Faliszewski et al. [13]; we do not consider p values between 0 and 1).

Computing HUV Committees. Unfortunately, for each $p > 0$ it is NP-hard to tell if there is a committee with at least a given score under the p-HUV rule [3, 33] and, as a consequence, no polynomial-time algorithms are known for these rules (for 0-HUV it suffices to sort the candidates in terms of their total utilities and, up to tie-breaking, take top k ones). We consider two ways of circumventing this issue:

(1) We use the standard greedy algorithm: To compute a p-HUV committee of size-k (for some $p > 0$), we start with an empty committee and perform k iterations, where in each iteration we extend the committee with a single resource that maximizes its p-HUV score. A classic result on submodular optimization shows that the committees computed this way are guaranteed to achieve at least $1 - 1/e$ fraction of the highest possible score [24].
(2) We use the simulated annealing heuristic, as implemented in the *simanneal* library, version 0.5.0. We set the number of steps to 50 000 and the temperature to vary between 9900 and 0.6.

In principle, we could also use the formulations of p-HUV rules as integer linear programs (ILPs), provided, e.g., by Skowron et al. [33] and Peters and Lackner [26]. Yet, given the sizes of our elections this would be quite infeasible (e.g., for the movie *Alice in Wonderland (1951)* we obtain an election with 32'783 resources and 5'339 agents).

3 SYSTEM DESIGN

Let us now describe our voting-based search system. The main idea is that for a *query set* of resources (such as a set of movies that someone enjoys) we form an election that regards related resources and whose winning committee is our *result*. Depending on the voting rule used, the result may contain resources either very closely or only somewhat loosely connected to those from the query. The system consists of three main components, the data model, the search model, and voting.

3.1 Data Model

The data model is responsible for converting domain-specific raw data into what we call a *global (approval) election*. For example, the raw data may consist of information how various people rate movies, or what products they buy in some store, or it may be generated using some statistical model of preferences (which, indeed, we will do to test our system in a controlled environment).

The global election stores our full knowledge of the domain. The interpretation is that the agents are the users who have interacted with some resources and they approve those for which the interaction was positive (for example, if they enjoyed a particular movie). Lack of an approval either means that the interaction was negative or that there was no interaction (while we could distinguish these two cases, we find using basic approval elections to be simpler).

The algorithm for forming the global election is the only domain-specific part of our model. Below we provide an example how such an algorithm may work.

Example 3.1. Consider the MovieLens 25M dataset [16]. It contains 25'000'095 ratings of 62'423 movies, provided by 162'541 users (so, on average, each user rated almost 154 movies). Each rating is on the scale from one to five stars (the higher, the better) and was provided between January of 1995 and November of 2019 on the MovieLens website. We form a global election where each user is an agent, each movie is a resource, and a user approves a movie if he or she gave it at least four stars. We remove from consideration those movies that were approved by fewer than 20 agents.

3.2 Search Model

The search model is responsible for forming a *local (utility) election*, specific to a particular query. The idea is that this election's winning committees would form our result sets. We first form a local approval election and then, if desired, we derive more fine-grained utilities for the agents.

Let $E = (R, A)$ be the global approval election, where $A = (a_1, \ldots, a_n)$, and let $Q \subseteq R$ be the query set. Let $N = \{1, \ldots, n\}$ be the set of agents present in E and let N_{local} be a subset of N containing those agents who approve at least one member of Q (while we could use other criteria, we focus on singleton query sets for which it is not relevant). Then, let R_{local} consist of those resources that are approved by the agents from N_{local}, except those from the query:

$$R_{local} = \{r \in R \mid (\exists i \in N_{local})[a_i(r) = 1]\} \setminus Q.$$

Finally, let A_{local} be the approval profile of the agents from N_{local}, restricted to the resources from R_{local}, and let $E_{local} = (R_{local}, A_{local})$ be our local approval election. Intuitively, it contains the knowledge about exactly those resources that were appealing to (some of) the agents that also enjoyed members of Q. Unfortunately, as shown below, it may be insufficient to provide relevant search results.

Example 3.2. Consider the MovieLens global election from Example 3.1 and let the query set Q consist of a single movie, *Hot Shots!* (a 1991 parody of the *Top Gun (1986)* movie, full of quirky/absurd humor). The five most-approved movies in the local approval election for Q are: (1) *The Matrix (1999)*, (2) *Back to the Future (1985)*, (3) *Fight Club (1999)*, (4) *Pulp Fiction (1994)*, and (5) *Lord of the Rings: The Fellowship of the Ring (2001)*. This is also the winning committee under the AV rule with $k = 5$. Neither of these movies has much to do with *Hot Shots!*, and they were selected because they are globally very popular (indeed, we expect many more readers of this paper to have heard of these five movies than of the one from the query set). Such globally popular movies are also popular among people enjoying *Hot Shots!*.

To address the above issue, we derive a local utility election $E_{util} = (R_{util}, U_{util})$, where $R_{util} = R_{local}$, which promotes those resources that are more particular to a given query. To this end, we use the *term frequency-inverse document frequency (TF-IDF)* mechanism.

Table 1: Results provided by our system for the movie *Hot Shots!* (see Examples 3.1, 3.4, and 3.6).

	exact algorithm	simulated annealing		greedy algorithm	
#	0-HUV	1-HUV	2-HUV	1-HUV	2-HUV
1.	The Naked Gun 2 1/2 (1991)	Hot Shots! Part Deux (1993)	Hot Shots! Part Deux (1993)	The Naked Gun 2 1/2 (1991)	The Naked Gun 2 1/2 (1991)
2.	Hot Shots! Part Deux (1993)	The Loaded Weapon 1 (1993)	The Loaded Weapon 1 (1993)	Hot Shots! Part Deux (1993)	The Loaded Weapon 1 (1993)
3.	Top Secret (1984)	The Naked Gun 2 1/2 (1991)	The Villain (1979)	The Loaded Weapon 1 (1993)	Major League II (1994)
4.	The Naked Gun (1988)	Cannonball Run II (1984)	Top Secret (1984)	Major League II (1994)	Yamakasi (2001)
5.	The Loaded Weapon 1 (1993)	Top Secret (1984)	Ernest Goes to Jail (1990)	Top Secret (1984)	Hot Shots! Part Deux (1993)
6.	Police Academy (1984)	Nothing to Lose (1997)	Last Boy Scout, The (1991)	Yamakasi (2001)	To Be or Not to Be (1983)
7.	The Last Boy Scout (1991)	Dragnet (1987)	Dragnet x(1987)	Hudson Hawk (1991)	Hudson Hawk (1991)
8.	Commando (1985)	Major League II (1994)	Freaked (1993)	To Be or Not to Be (1983)	Freaked (1993)
9.	Hudson Hawk (1991)	Yamakasi (2001)	Major League II (1994)	City of Violence (2006)	Top Secret (1984)
10.	Twins (1988)	Coffee Town (2013)	Yamakasi (2001)	Dragnet (1987)	City of Violence (2006)

TF-IDF. This is a standard heuristic introduced by Jones [17, 18] to evaluate how specific is a given term t for a document d from a document corpus D (for further information on TF-IDF see, e.g., the works of Robertson and Walker [31] and Ounis [25]). The main idea is that the specificity value of t in d is proportional to the frequency of t in d (term frequency; *TF*) and inversely proportional to the frequency of t in all the documents D (inverse document frequency; *IDF*).

Given our global election $E = (R, A)$ and the approval local election $E_{local} = (R_{local}, A_{local})$, we implement the TF-IDF idea as follows. We interpret the resources as the terms, and we take the document corpus to consist of two "documents," election E_{local} and election $E' = (R_{local}, A')$, where A' is the approval profile for those agents from the global election that do not appear in E_{local}. Let n be the total number of agents. For a resource $r \in R_{local}$, we let its *term frequency* component be the number of agents that approve it in the local election, i.e., $\mathrm{tf}(r) = |A_{local}(r)|$. We let r's inverse document frequency be $\mathrm{idf}(r) = \ln\left(n/|A(r)|\right)$. Finally, to balance the TF and IDF components, we assume we have some constant γ and we define:

$$\text{tf-idf}_\gamma(r) = \mathrm{tf}(r)\gamma^{\mathrm{idf}(r)} = \frac{|A_{local}(r)|}{A(r)^{\ln\gamma}} \cdot \left(n^{\ln\gamma}\right).$$

Example 3.3. Consider three resources, r_1, r_2, and r_3, where: $|A_{local}(r_1)| = 1$, $|A(r_1)| = 2$, $|A_{local}(r_2)| = 10$, $|A(r_2)| = 20$, and $|A_{local}(r_3)| = 100$, $|A(r_3)| = 2000$. If we focused on the number of approvals in the local election (by taking $\ln\gamma = 0$), then we would view r_3 as the most relevant resource. This would be unintuitive as only a small fraction of r_3's approvals come from the agents who enjoy the items in the query set. For $\gamma \approx 2.78$ (i.e., $\ln\gamma = 1$), we would focus on the ratios $|A'(r_i)|/|A(r_i)|$, so r_1 and r_2 would be equally relevant, and r_3 would come third. This is more appealing, but still unsatisfying as, generally, r_2 is more popular than r_1. By taking, e.g., $\gamma = 2$ (i.e., $\ln\gamma \approx 0.69$) we would focus on ratios $|A'(r_i)|/|A(r_i)|^{0.69}$ and, indeed, r_2 would be the most relevant resource, followed by r_1 and r_2.

We have found that $\gamma = 2$ works best for our scenario (we discuss the process of choosing this value in Section 4.1).

Example 3.4. Consider the same setting as in Example 3.2, but take five movies with the highest TF-IDF value (for $\gamma = 2$). We obtain: (1) *The Naked Gun 2 1/2 (1991)*, (2) *Hot Shots! Part Deux (1993)*, (3) *Top Secret! (1984)*, (4) *The Naked Gun (1988)*, (5) *The Loaded Weapon 1 (1993)*. All these movies are parodies similar in style to *Hot Shots!*.

Local Utility Election. We form the local utility election $E_{util} = (R_{util}, U_{util})$ by setting the utilities as follows. Given an agent i from the local approval election and a resource $r \in R_{util}$, if agent i approves r, then set $u_i(r) = \text{tf-idf}(r)/|A_{local}(r)|$. Otherwise, set it to be 0. This way the utilities assigned to a given resource sum up to its TF-IDF value.

Example 3.5. By the design of the local utility election, a committee of size $k = 5$ for the *Hot Shots!* local utility election would consist exactly of the five movies listed in Example 3.4.

3.3 Voting

The last component of our system is to compute (an approximation of) a winning committee under the local utility election under a given p-HUV rule. If we are looking for resources that are the most closely connected to the query set, then we take $p = 0$. For a broader search, we typically consider $p \in \{1, 2, 3, \ldots\}$. Since most of our rules are NP-hard to compute, we compute our committees either using the greedy approximation algorithm or simulated annealing. The greedy algorithm returns the committee ordered with respect to the iteration number in which a given resource was added (thus, the first resource is always the same for a given election, irrespective of p). The algorithm based on simulated annealing outputs the committee in an arbitrary order.

Example 3.6. Consider the local utility election for the *Hot Shots!* movie. In Table 1 we show the p-HUV committees for, $p \in \{0, 1, 2\}$, where for 0-HUV we use the exact algorithm and for the other two rules we use simulated annealing and the greedy algorithm. Let us discuss the contents of these committees (for $p \in \{1, 2\}$, we focus on simulated annealing):

(1) The first six movies selected by 0-HUV are quirky, absurd comedies, quite in spirit of *Hot Shots!*; among the next four movies, three are comedies (one of which is somewhat similar in spirit to the first six) and one is an action movie.
(2) Except for *Yamakasi*, all movies selected by 1-HUV are comedies of different styles, including four of the same style as *Hot Shots!*, two action comedies, two family comedies, and one crime comedy. *Yamakasi* is an action/drama movie which stands out from the rest.

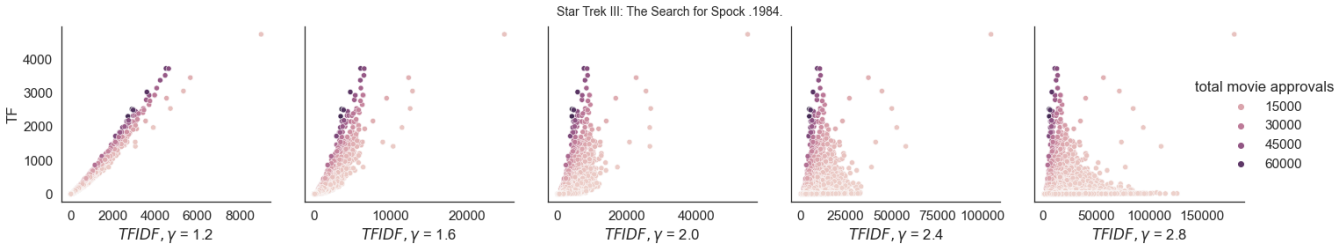


Figure 1: Each dot represents a movie in the local election generated for *Star Trek III: The Search for Spock (1984)* and shows the relation between the movie's popularity (its *TF* value) on the y axis and its final TF-IDF score on the x axis for various setting of the parameter γ. The hue of the dot represents the popularity of the movie in the global election (i.e., the number of its approvals)

(3) 2-HUV selects even more varied set of comedies than 1-HUV, including a western comedy, a sci-fi comedy, crime/action comedies, and family movies. Yet, it also includes *Yamakasi*.

The reason why *Yamakasi* is included in our 1-HUV and 2-HUV committees is simply because, in total, it only received 53 approvals, of which 27 came from people who enjoyed *Hot Shots!*. Thus it was viewed as a very relevant movie for the query. If we replaced the simple TF-IDF heuristic with a more involved scoring system (possibly using more information about the movies), we could account for such situations better.

The committees computed by the greedy algorithm for 1-HUV and 2-HUV are of comparable quality to those provided by simulated annealing, although they include eight comedies and two action movies each.

4 EXPERIMENTS

In this section we present three experiments. The first and the last one are conducted on the MovieLens dataset, whereas the middle one uses synthetic data. In the first experiment, we describe our process of choosing the γ value for the TF-IDF heuristic. In the second one, we test if, indeed, 0-HUV rule focuses on resources very similar to the one from the query set, whereas p-HUV rules for $p \in \{1, 2, 3\}$ seek increasingly more diverse result sets. In the final experiment, we compare the performances of the greedy algorithm and simulated annealing.

4.1 Calibrating the TF-IDF Metric

Before we describe our formal procedure for choosing the γ parameter, let us explore its meaning. Intuitively, γ is used to give more weight to the IDF component relative to the TF one. In other words, replacing γ with a larger value more strongly diminishes the TF-IDF values of the globally more popular movies than of the less popular ones (i.e., those with fewer approvals in the global election). This balance is visualised in Figure 1, where we consider the movie *Star Trek III: The Search for Spock (1984)*, and for each movie in the local approval election we draw a dot showing the relation between its number of approvals in the local election (i.e., its TF value), on the y axis, and its final TF-IDF value, on the x axis, for several values of γ. The hue of the dot represents the number of approvals of the movie in the global election. The top ten movies according to TF-IDF (for a given γ) are the ten rightmost dots in the respective diagram. Note that the higher the γ is, the more dots with low TF value appear to the right and have higher chance of being among top ten movies. For $\gamma = 1.2$, quite a few generally popular items (with darker hue) make it to the top ten, simply because they are popular overall and not only in the context of the search for a given query set. For $\gamma = 2.0$, there seems to be a good balance between the popular and not so popular movies, while for $\gamma = 2.8$ there are only extremely unpopular movies selected for the top ten.

The above argument for using $\gamma = 2$ is based on intuition and to get a better grounding for the choice, we have performed the following experiment. Our basic premise is that the γ value should be such than when searching for a query set consisting of a single movie, its most similar movies should appear among the top ten, with respect to the TF-IDF metric. While deciding what "the most similar movie" is is quite a subjective issue, we assumed that for all the movies from the *Star Trek* series, other *Star Trek* movies are the most similar ones. The MovieLens dataset contains fourteen *Star Trek* movies (that are approved by at least 20 users) and we list them, together with their approval counts in the global election, in Table 2.

Table 2: All the Star Trek movies present in the MovieLens 25M dataset (with at least 20 approvals), together with their approval counts.

Movie	Approval Count
Star Trek: Renegades	27
Star Trek: Nemesis	1904
Star Trek Beyond	1987
Star Trek V: The Final Frontier	2338
Star Trek: Insurrection	3783
Star Trek: The Motion Picture	3785
Star Trek III: The Search for Spock	4732
Star Trek VI: The Undiscovered Country	5358
Star Trek Into Darkness	5621
Star Trek IV: The Voyage Home	7544
Star Trek II: The Wrath of Khan	10669
Star Trek: Generations	10809
Star Trek: First Contact	12396
Star Trek	12854

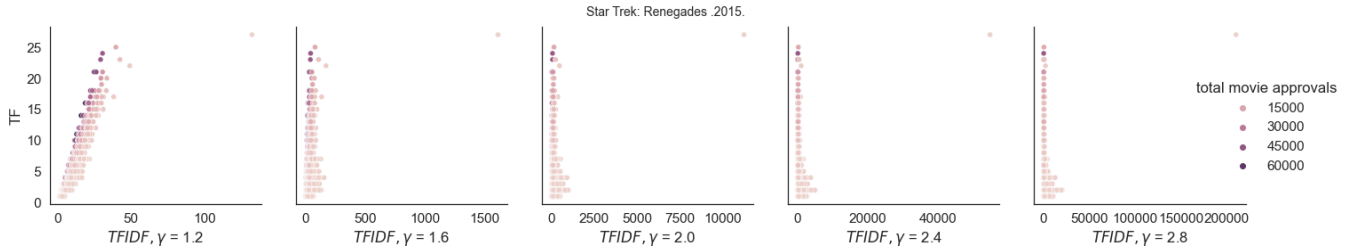


Figure 2: Analogous to Figure 1 but for the movie *Star Trek: Renegades (2015).*

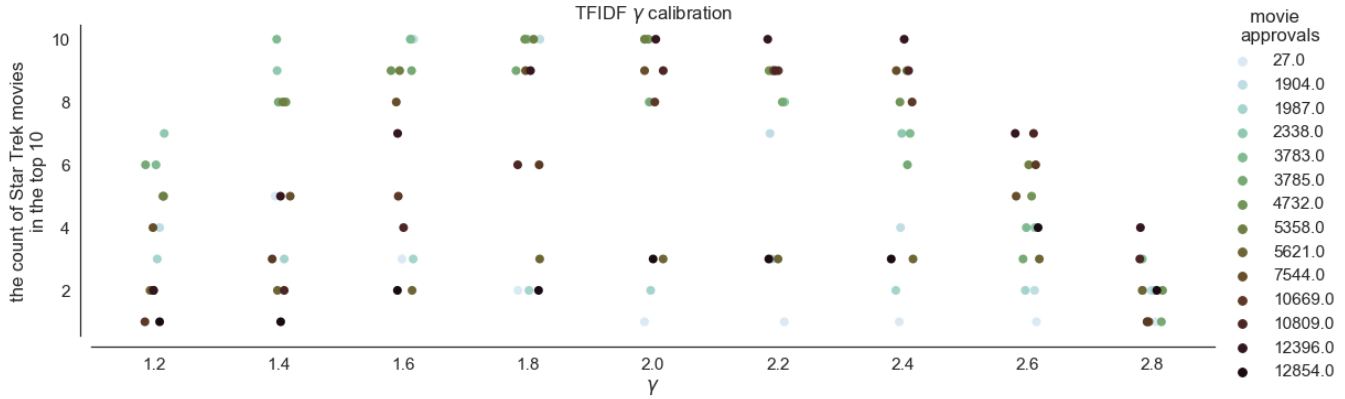


Figure 3: The calibration experiment results. Each dot represents the *top 10 count* value (vertical axis) when searching for a single movie (hue represents the popularity of the movie, the darker the more approvals the movie has) using TF-IDF with a given γ (horizontal axis).

We used each *Star Trek* movie as a singleton query set, computed its local approval election, ranked the movies from this election with respect to their TF-IDF values for $\gamma \in \{1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8\}$, and, for each of these values of γ, calculated how many *Star Trek* movies are among the top ten ones. We present the results in Figure 3, where each dot corresponds to a *Star Trek* movie and the values on the y axis specify how many other *Star Trek* movies were among top-ten with respect to TF-IDF and the given value of γ. Finally, we have averaged these values for each γ over all the fourteen movies (see Table 3). It turned out that $\gamma = 2$ indeed gave the best results (naturally, doing a more fine-grained search for the value of γ might lead to a slightly different outcome, but we did not feel it would affect the other results in the paper significantly).

While setting $\gamma = 2$ works well for many movies, it is not as good a choice for some others. Consider Figure 2 which is analogous to Figure 1, but for the movie *Star Trek: Renegades (2015)*, which is not very popular and its local approval election contains relatively few movies (dots on the diagram). Note that taking $\gamma = 2$ does not seem to strike the right balance and we would need to choose γ between 1.2 and 1.6. Nevertheless, we choose the simplicity of choosing a single value of the γ parameter for all the movies, at the cost of getting a few of them wrong.

Table 3: Star Trek TF-IDF calibration results. The relation between γ and the average number of Star Trek movies found in the top 10 results according to TF-IDF when searching for all the Star Trek movies from MovieLens 25M dataset.

γ	Average top ten count
1.2	3.53
1.4	5.13
1.6	6.07
1.8	6.53
2.0	**6.73**
2.2	6.47
2.4	5.73
2.6	3.93
2.8	1.80

4.2 Synthetic Data: Focus Versus Breadth

In the second experiment we generate the global election synthetically, so that it resembles data regarding movies. The point is to observe the differences between committees computed according to p-HUV rules for different values of p in a controlled environment.

Generating Global Elections. We assume that we have nine main categories of movies (such as, e.g., a comedy or a thriller) and each category has nine subcategories (such as, e.g., a romantic comedy, or a psychological thriller). For each pair of a category $u \in [9]$ and its subcategory $v \in [9]$, we generate 25 movies, denoted

$u.v(1), \ldots, u.v(25)$. Given a movie $u.v(i)$, we set its *quality factor* to be $q(i) = -\arctan\frac{i-13}{10} + 2$. That is, for each subcategory the first movie has the highest quality, about 2.87, and the qualities of the following movies decrease fairly linearly, down to about 1.12. We also have $n = 2000$ voters. Each voter i has a probability distribution P_i over the main categories (where we interpret $P_i(u)$ as the probability that the voter watches a movie from category u), and for each category u, he or she has probability distribution P_i^u over its subcategories (interpreted as the conditional probability that if the voter watches a movie from category u, then this movie's subcategory is v). For each voter, we choose these distributions as permutations of:

$$(0.5, 0.1.0.1.0.1, 0.1, 0.025, 0.025, 0.025, 0.025)$$

chosen uniformly at random. Intuitively, each voter has his or her most preferred category, four categories that he or she also quite enjoys, and four categories that he or she rarely enjoys (the same applies to subcategories). To generate an approval of a voter i we do as follows:

(1) We choose a category u according to distribution P_i and, then, a subcategory v according to distribution P_i^u.
(2) We choose a movie among $u.v(1), \ldots, u.v(25)$ with probability proportional to its quality factor. The voter approves the selected movie.

We repeat this process 162 times for each voter, leading to a bit fewer approvals (due to repetitions in sampling; recall that in MovieLens the average number of approvals is 154).

While the above-described process of generating a global election is certainly quite ad-hoc, we believe that it captures some of the main features of people's preferences regarding movies. More importantly, we can say that two movies are very similar if they come from the same subcategory, are somewhat similar if they come from the same category but different subcategories, and are loosely related otherwise.

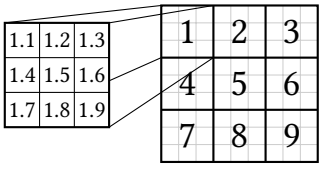


Figure 7: Visual arrangement of the movie categories and subcategories for the synthetic experiment.

Running The Experiment. For each number $p \in \{0, 1, 2, 3\}$ and both algorithms for computing approximate p-HUV committee (i.e., the greedy algorithm and simulated annealing) we repeat the following experiment.[1] We generate 100 global elections as described above, and for each of them we compute a committee of size $k = 10$ for the query set consisting of movie 1.1(13), i.e., the middle-quality movie from subcategory 1.1 (since the (sub)categories are, effectively, symmetric, their choice is irrelevant). Altogether, 1000 movies are selected (some of the movies are selected more than once and we count each of them). Then, for each subcategory $u.v$, we sum up, over all the computed committees, how many movies from this subcategory were selected, obtaining a histogram (in this respect, our experiment is quite similar to that of Elkind et al. [10]).

To present these histograms visually, we arrange the categories into a 3×3 square, where each category is further represented as a 3×3 subsquare of subcategories, as shown in Figure 7. We show thus-arranged histograms for simulated annealing in Figure 4 and for the greedy algorithm in Figure 5. Each subcategory square is labeled with the number of movies selected from this subcategory and its background reflects this number (darker backgrounds correspond to higher numbers). Further, in both figures next to the name of each p-HUV rule we report a vector (x, y, z), where x means the number of movies selected from subcategory 1.1, y means the number of movies selected from category 1 except for those in subcategory 1.1, and z refers to the number of all the other selected movies. Thus we always have that $x + y + z = 1000$.

Analysis. Our main conclusion is that, indeed, 0-HUV focuses on very similar movies (almost all the selected movies come from category 1.1) and as p increases, approximate p-HUV committees include more and more movies from other subcategories of category 1, and, eventually, even more movies outside of it. It would be desirable to have a value of p for which we would get a vector (x, y, z) close to, say, $(450, 450, 100)$, meaning that, on average, the resulting committee would contain between 4 and 5 movies from category 1.1 (i.e., directly relevant to the search), between 4 and 5 movies from other subcategories of category 1 (i.e., similar but quite different from the query), and 1 movie from some other category (i.e., something very different, but possibly appealing to the people who enjoyed the movie from the query). Yet, our algorithms do not seem to provide committees with such vectors (this, however, is not a major worry—after all, the setup in the experiment was simplified and, to some extent, Example 3.6 shows that for real-life MovieLens data we do find such committees).

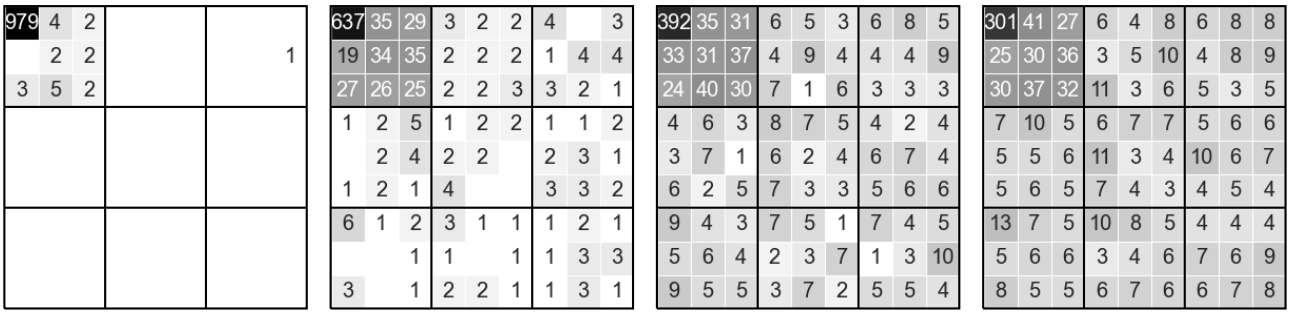
4.3 Effectiveness of the Approximations

In the final experiment we compare the quality of the committees computed by the greedy algorithm and by simulated annealing, for the MovieLens dataset. To this end, we sampled 100 movies and for each we have computed approximate winning committees for 1-HUV, 2-HUV, and 3-HUV, using both our algorithms. For each movie we calculated the ratio of the score computed using the greedy algorithm and simulated annealing. We show the results in Figure 6 and, in a more aggregate form, in Table 4. On average, the greedy algorithm finds committees with about $2 - 3\%$ higher scores than simulated annealing (especially for the more popular movies). Yet, as we have seen in Example 3.6, simulated annealing has other positive features. Yet, we note that we ran the simulated annealing algorithm for 50000 steps and using more would certainly improve the result (yet, simulated annealing already takes four times as long to compute as the greedy algorithm).

Table 4: Average ratios of committee scores computed with the greedy algorithm and with simulated annealing.

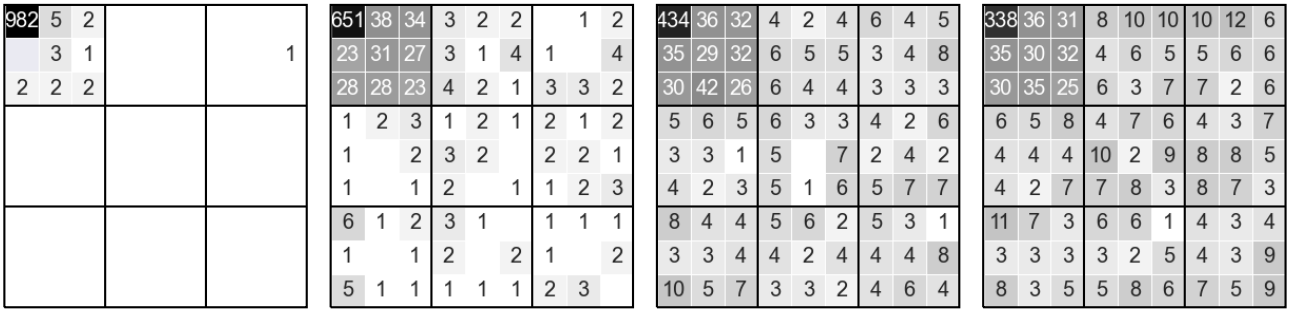
rule	Mean	Std. Dev.
1-HUV	1.021	0.015
2-HUV	1.027	0.021
3-HUV	1.034	0.028

[1] We compute the 0-HUV committees using the optimal, polynomial-time algorithm.



(a) 0-HUV, (979, 20, 1) **(b)** 1-HUV, (637, 230, 133) **(c)** 2-HUV, (392, 261, 347) **(d)** 3-HUV, (301, 258, 441)

Figure 4: Histograms for the synthetic experiment and simulated annealing.



(a) 0-HUV, (982, 17, 1) **(b)** 1-HUV, (651, 232, 117) **(c)** 2-HUV (434, 262, 304) **(d)** 3-HUV (338, 254, 408)

Figure 5: Histograms for the synthetic experiment and the greedy algorithm.

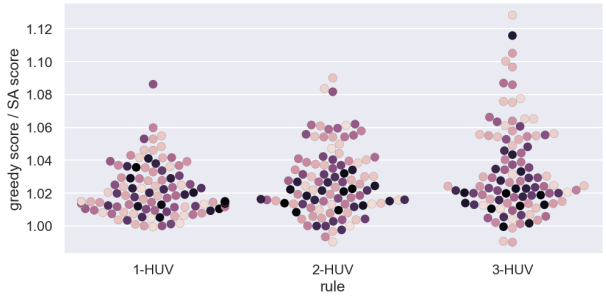


Figure 6: Effectiveness of the greedy algorithm versus simulated annealing (see Section 2 for details of the algorithm). Each dot represents a single movie (from a set of 100 randomly selected ones), and is used as a singleton query set. Its position on the y axis is the ratio of the scores of the committees computed for this movie using the greedy algorithm and simulated annealing. The position on the x-axis is perturbed to show all the dots. The color of the dot represents the number of approvals of the movie in the global election (the darker it is, the more approvals.

5 CONCLUSIONS

We have shown that multiwinner voting can be successfully used to build a system that helps searching for movies and that lets the users specify how strongly related should the proposed set of movies be to those he or she asks about. Our system does not use any advanced tools for non-personalized recommendation systems and purely demonstrates that multiwinner voting is something that designers of such systems might want to consider as a tool.

We do not discuss running times of our algorithms. They are implemented using Python and are highly non-optimized, so analyzing their running times would not be very meaningful.

ACKNOWLEDGMENTS

Grzegorz Gawron was supported in part by AGH University of Science and Technology and the "Doktorat Wdrożeniowy" program of the Polish Ministry of Science and Higher Education. The presented research contains preliminary ideas that lead to the proposal of the PRAGMA project (currently funded by the European Research Council, grant agreement No. 101002854).

REFERENCES

[1] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. 2013. Movie Recommender System for Profit Maximization. In *Proceedings of RecSys-13*. 121–128.

[2] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, and T. Walsh. 2017. Justified Representation in Approval-Based Committee Voting. *Social Choice and Welfare*

48, 2 (2017), 461–485.
[3] H. Aziz, S. Gaspers, J. Gudmundsson, S. Mackenzie, N. Mattei, and T. Walsh. 2015. Computational Aspects of Multi-Winner Approval Voting. In *Proceedings of AAMAS-15*. 107–115.
[4] N. Betzler, A. Slinko, and J. Uhlmann. 2013. On the Computation of Fully Proportional Representation. *Journal of Artificial Intelligence Research* 47 (2013), 475–519.
[5] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, D. Knop, and R. Niedermeier. 2020. Parameterized Algorithms for Finding a Collective Set of Items. In *Proceedings of AAAI-20*. 1838–1845.
[6] M. Brill, J.-F. Laslier, and P. Skowron. 2018. Multiwinner Approval Rules as Apportionment Methods. *Journal of Theoretical Politics* 30, 3 (2018), 358–382.
[7] A. Chakraborty, G. Patro, N. Ganguly, K. Gummadi, and P. Loiseau. 2019. Equality of Voice: Towards Fair Representation in Crowdsourced Top-K Recommendations. In *Proceedings of FAT-19*. 129–138.
[8] B. Chamberlin and P. Courant. 1983. Representative Deliberations and Representative Decisions: Proportional Representation and the Borda Rule. *American Political Science Review* 77, 3 (1983), 718–733.
[9] S. Choi, S. Ko, and Y. Han. 2012. A Movie Recommendation Algorithm Based On Genre Correlations. *Expert Systems with Applications* 39, 9 (2012), 8079–8085.
[10] E. Elkind, P. Faliszewski, J. Laslier, P. Skowron, A. Slinko, and N. Talmon. 2017. What Do Multiwinner Voting Rules Do? An Experiment Over the Two-Dimensional Euclidean Domain. In *Proceedings of AAAI-17*. 494–501.
[11] E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. 2017. Properties of Multiwinner Voting Rules. *Social Choice and Welfare* 48, 3 (2017), 599–632.
[12] P. Faliszewski, J. Sawicki, R. Schaefer, and M. Smolka. 2017. Multiwinner Voting in Genetic Algorithms. *IEEE Intelligent Systems* 32, 1 (2017), 40–48.
[13] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. 2017. Multiwinner Rules on Paths From k-Borda to Chamberlin–Courant. In *Proceedings of IJCAI-17*. 192–198.
[14] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. 2017. Multiwinner Voting: A New Challenge for Social Choice Theory. In *Trends in Computational Social Choice*, U. Endriss (Ed.). AI Access Foundation.
[15] Sumit Ghosh, Manisha Mundhe, Karina Hernandez, and Sandip Sen. 1999. Voting for movies: the anatomy of a recommender system. In *Proceedings of the third annual conference on Autonomous Agents*. 434–435.
[16] F. M. Harper and J. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19:1–19:19.
[17] K. Spärck Jones. 1973. Index Term Weighting. *Information Storage and Retrieval* 9, 11 (1973), 619–633.
[18] K. Spärck Jones. 2004. A Statistical Interpretation of Term Specificity And Its Application In Retrieval. *Journal of Documentation* 60, 5 (2004), 493–502.
[19] Y. Kim, K. Kim, C. Park, and H. Yu. 2019. Sequential and Diverse Recommendation with Long Tail. In *Proceedings of IJCAI-19*. 2740–2746.
[20] M. Lackner and P. Skowron. 2018. Consistent Approval-Based Multi-Winner Rules. In *Proceedings of EC-18*. 47–48.
[21] M. Lackner and P. Skowron. 2020. *Approval-Based Committee Voting: Axioms, Algorithms, and Applications*. Technical Report arXiv:2007.01795 [cs.GT]. arXiv.org.
[22] M. Lackner and P. Skowron. 2020. Utilitarian Welfare and Representation Guarantees of Approval-Based Multiwinner Rules. *Artificial Intelligence* 288 (2020), 103366.
[23] A. Mondal, R. Bal, S. Sinha, and G. Patro. 2020. *Two-Sided Fairness in Non-Personalised Recommendations*. Technical Report arXiv:2011.05287 [cs.AI]. arXiv.org.
[24] G. Nemhauser, L. Wolsey, and M. Fisher. 1978. An Analysis of Approximations for Maximizing Submodular Set Functions. *Mathematical Programming* 14, 1 (Dec. 1978), 265–294.
[25] I. Ounis. 2018. Inverse Document Frequency. In *Encyclopedia of Database Systems*. Springer.
[26] D. Peters and M. Lackner. 2020. Preferences Single-Peaked on a Circle. *Journal of Artificial Intelligence Research* 68 (2020), 463–502.
[27] V. Phonexay, D. Park, K. Xinchang, and F. Hao. 2018. An Efficient Movie Recommendation Algorithm Based On Improved k-Clique. *Human-centric Computing and Information Sciences* 8 (2018), 38.
[28] M. Pourghanbar, M. Kelarestaghi, and F. Eshghi. 2015. EVEBO: A New Election Inspired Optimization Algorithm. In *Proceedings of CEC-15*. 916–924.
[29] A. Procaccia, J. Rosenschein, and A. Zohar. 2008. On the Complexity of Achieving Proportional Representation. *Social Choice and Welfare* 30, 3 (2008), 353–362.
[30] F. Ricci, L. Rokach, and B. Shapira (Eds.). 2015. *Recommender Systems Handbook*. Springer.
[31] S. Robertson and S. Walker. 1997. On Relevance Weights with Little Relevance Information. In *Proceedings of SIGIR-97*. 16–24.
[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of WWW-01*. 285–295.
[33] P. Skowron, P. Faliszewski, and J. Lang. 2016. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence* 241 (2016), 191–216.
[34] T. Thiele. 1895. Om Flerfoldsvalg. In *Oversigt over det Kongelige Danske Videnskabernes Selskabs Forhandlinger*. 415–441.
[35] R. Yager. 1988. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics* 18, 1 (1988), 183–190.