DVote: Constraining Committee Voting with Database Dependencies

Roi Yona
Technion
Haifa, Israel
roi.yona@campus.technion.ac.il

Jonathan Breitman
Technion
Haifa, Israel
jonathanbre@campus.technion.ac.il

Benny Kimelfeld
Technion & RelationalAI
Haifa, Israel
bennyk@cs.technion.ac.il

ABSTRACT

Approval-Based Committee (ABC) voting refers to the task of selecting a committee of a desired size, given voter preferences that state the specific candidates that each voter approves of. A voting rule aggregates the voter preferences into a winning committee. As a special case, an ABC scoring rule determines a score that each voter contributes to the committee based on her approvals. Various ways have been proposed to impose constraints on the elected committee. The demonstration presents DVote— a tool that implements a recent framework for extending score-based ABC voting with constraints on the context surrounding the candidates, given as a relational database. DVote provides a convenient interface to set up a voting instance and build contextual constraints in the form of Tuple-Generating Dependencies (TGDs) and Denial Constraints (DCs). The computation of the winning committee is done by backend components that encapsulate the contextual database and translate the entire task of constrained election into Mixed Integer Programming. In the demonstration, attendees will experience ABC voting with DVote in different domains and contexts.

PVLDB Reference Format:
Roi Yona, Jonathan Breitman, and Benny Kimelfeld. DVote: Constraining Committee Voting with Database Dependencies. PVLDB, 18(12): 5235 - 5238, 2025.
doi:10.14778/3750601.3750640

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Roi-Yona/abcc.git.

1 INTRODUCTION

In Approval-Based Committee (ABC) voting, a voter preference is provided as a set of candidates approved by the voter, and voter approvals are aggregated into a choice of a winning committee [8]. The manner of aggregation is called an *ABC voting rule*. This work focuses on a well-studied class of ABC voting rules, namely the ABC *scoring rules*: every voter contributes a score to each candidate set, based on two numbers: the number of approved candidates in the set, and the total number of candidates that the voter approves of; a winning committee (of the desired size) is one with the highest accumulated score [9]. Yet, it is commonly the case that the elected committee should also satisfy criteria beyond having the highest voter support. For example, it may need to feature a set of skills so that it can properly function, avoid conflicts among members, provide sufficient representation of issues, avoid over-representation of issues, and so on. Hence, a significant body of work studied ABC voting (and multi-winner elections in general) in the presence of constraints. Various formalisms have been proposed for such constraints, like cardinality constraints over occurrences of labels of the candidates [5], graphs of conflicts among candidates [11], and proportionality axioms with respect to voter representation [1].

Committee constraints are derived from contextual knowledge about the candidates, such as different types of relationships between candidates and associated entities, or among the candidates themselves. Inspired by previous frameworks that connect computational social choice and databases [6, 7], we recently proposed and investigated a framework for constraint languages over ABC voting in the presence of contextual information [12]. Specifically, the context is a relational database conforming to a relational context schema. We identify contextual constraints as database dependencies over the context schema extended with a virtual relation (namely, Com) that represents the committee. Given an ABC scoring rule, a winning committee is a set of candidates with a maximum score such that all database dependencies hold in the extended database. Hence, the framework allows us to deploy the rich literature of database dependencies toward expressive and intuitive languages for constraining ABC voting.

The demonstration presents DVote—a tool that encapsulates our framework in an end-to-end application for ABC voting in the presence of user-provided constraints over an arbitrary context database of choice [12]. We focus on two types of database dependencies: Tuple-Generating Dependencies (TGDs) [3] and Denial Constraints (DCs) [4]. The former can express, for example, representations of groups (conditional on the representation of other groups), while the latter can express conflicts among candidates. While the selection of a winning committee (or even just determining whether any legal committee exists) is NP-hard, we showed its feasibility via Mixed Integer Programming (MIP) that applies to arbitrary ABC scoring rules, TGDs and DCs. We also developed techniques for optimizing the MIP. See the full paper [12] for the empirical study that shows the effectiveness of the implementation.

DVote provides a convenient interface to choose the ABC settings (e.g., the committee size and ABC scoring rule), easily build TGDs and DCs over the extended context schema, execute the actual election, and reiterate the process until a satisfactory committee is established. The backend includes the context database, our algorithms for building the MIP from the database and constraints,

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750640

Companion video available at https://www.youtube.com/watch?v=pgQBm8xRQuU.

and the MIP solver. The demonstration scenarios involve several election domains (e.g., political elections, movies, and hotels) along with their context databases. The attendees of the demonstration interact with DVote by invoking elections, viewing the results, phrasing different types of constraints, and observing their impact on the outcome of the modified election setup.

2 ABC VOTING WITH CONTEXTUAL CONSTRAINTS

We first describe shortly the framework of ABC voting with contextual constraints. See the full paper [12] for more details, theoretical complexity analysis, implementation description, and experiments.

ABC Voting. In Approval-Based Committee (ABC) voting, we have a set $C = \{c_1, \ldots, c_m\}$ of candidates, a set $V = \{v_1, \ldots, v_n\}$ of voters, an approval profile A that maps every voter v_i to a set $A(v_i) \subseteq C$ of candidates, and a desired committee size k. An *ABC voting rule* determines which k-subsets of C (i.e., sets of k candidates) are the winning committees. As a special case, an *ABC scoring rule* [9] is defined by a function $f(x, y)$ that determines the score contributed to a committee B by a voter v who approves a total of y candidates where x of them are in B. The winning committees by f are the k-subsets B that maximize

$$\text{score}_f(B) := \sum_{v \in V} f(|B \cap A(v)|, |A(v)|)\,.$$

The class of ABC scoring rules generalizes the class of *Thiele rules* where $f(x, y)$ is a non-decreasing function $w(x)$ that depends only on the number of approved candidates. Examples include *Approval Voting* (AV) where $w(x) = x$, Proportional Approval Voting (PAV) where $w(x) = \sum_{i=1}^{x} 1/i$, and Chamberlin-Courant (CC) where $w(x) = \min(x, 1)$ It has been established that finding a winning committee is computationally hard for all Thiele rules, except for AV where the problem is straightforward [2, 10].

Databases and dependencies. A *relational schema* $\mathcal{S}$ consists of a finite collection of *relation symbols* R, each associated with an arity k. A *database* D over the schema $\mathcal{S}$ associates a finite relation R^D of k-tuples to each k-ary relation symbol R. A *constraint* over $\mathcal{S}$ is a formula in some logical formalism (e.g., first-order logic) with relation symbols in $\mathcal{S}$. If Γ is a set of constraints, then $D \models \Gamma$ denotes that D satisfies every constraint in Γ. An *atomic formula* has the form $R(\tau_1, \ldots, \tau_k)$ or $\tau_1 \theta \tau_2$, where R is a k-ary relation symbol, each τ_i is either a constant or a variable, and θ is a predefined comparison operator (e.g., $=, \leq, >$ and so on). We call $R(\tau_1, \ldots, \tau_k)$ a *relational atom* and $\tau_1 \theta \tau_2$ a *comparison atom*.

A *denial constraint* (DC) is an expression of the form

$$\forall \vec{x}[\neg(\varphi(\vec{x}) \wedge \psi(\vec{x}))]$$

where $\vec{x}$ is a sequence of variables, $\varphi(\vec{x})$ is a conjunction of relational atoms, and $\psi(\vec{x})$ is a conjunction of comparison atoms [4].

A *tuple-generating dependency* (TGD) has the form $\forall \vec{x}[\varphi(\vec{x}) \rightarrow \exists \vec{y}[\psi(\vec{x}, \vec{y})]]$, where $\vec{x}$ and $\vec{y}$ are disjoint sequences of variables and $\varphi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ are conjunctions of relational atoms [3]. TGDs generalize common constraints like *inclusion constraints*; for example, in the database of Figure 1, the TGD

$$\forall x, y[\text{Author}(x, y) \rightarrow \exists z[\text{Pub}(y, z)]$$

Approvals:

v_1:	Ann, Dave	v_3:	Ann, Eva
v_2:	Ann, Bob, Dave	v_4:	Cale
		v_5:	Bob, Dave

Topic

name
AI
ML
OS
PL

Supervise

advisor	advised
Ann	Bob
Bob	Fred
Cale	Eva
Dave	Fred

Author

author	pub
Ann	p1
Ann	p2
Bob	p1
Bob	p3
Cale	p4
Dave	p5

Pub

pub	topic
p1	ML
p2	PL
p3	OS
p4	AI
p5	OS

Figure 1: ABC voting with external context.

requires every publication in the Author relation to occur in the Pub relation. Note that $\varphi(\vec{x})$ can be an empty conjunction, hence a tautology (meaning that $\exists \vec{y}[\psi(\vec{x}, \vec{y})]$ should hold unconditionally), and we denote such $\varphi(\vec{x})$ by **true**.

ABC voting with contextual constraints. We now recall the formal framework of ABC voting with contextual database and constraints [12]. Consider an approval profile A over a set C of candidates. We have information about the candidates in a database D over a schema $\mathcal{S}$. We view D as providing external context on the candidates. We wish to express constraints on the desired committee so that we restrict the class of eligible committees to those satisfying the constraints. We view the constraints as database dependencies on an *extended* schema $\mathcal{S}_+$, which is $\mathcal{S}$ augmented with a unary relation symbol Com that represents a hypothetical committee. By a *contextual constraint* we refer to a database dependency over $\mathcal{S}_+$. Given a database D and a set B of candidates, we write $D[B]$ to denote the database over $\mathcal{S}_+$ that consists of D and of the relation B interpreting the relation symbol Com. Thus, $R^{D[B]} := R^D$ for every relation symbol R of $\mathcal{S}$, and $\text{Com}^{D[B]} := B$.

An instance of our setting has the form $(V, C, A, k, f, \mathcal{S}, D, \Gamma)$ where (V, C, A, k, f) is an ordinary ABC setting with the objective of finding a committee of size k under the ABC scoring rule f; D is a database over the schema $\mathcal{S}$; and Γ is a set of constraints over the schema $\mathcal{S}_+$. A *legal committee* is a k-subset B of C such that $D[B] \models \Gamma$. A *winning legal committee* is a legal committee B such that $\text{score}_f(B)$ is the maximum among all legal committees B'.

Examples. In the following examples, we use $k = 3$ as the committee size and $f = \text{AV}$ as the scoring rule. The approval profile of Figure 1 (top) has $V = \{v_1, \ldots, v_5\}$ and $C = \{\texttt{Ann}, \texttt{Bob}, \texttt{Cale}, \texttt{Dave}, \texttt{Eva}\}$. Assume that we seek a Program Committee (PC) for a conference. The relations in the figure give information about candidate publications and advisory relationships. The following DC states that the PC cannot include both a person and the advisor of that person.

$$\forall c_1, c_2[\neg(\text{Supervise}(c_1, c_2) \wedge \text{Com}(c_1) \wedge \text{Com}(c_2))].$$

For this DC, the committee $\{\texttt{Ann}, \texttt{Bob}, \texttt{Dave}\}$ is illegal, since Ann is the advisor of Bob. Note that every set of three candidates (among the five) is a legal committee, as long as it does not include both Ann and Bob or both Cale and Eva. Here, $B = \{\texttt{Ann}, \texttt{Dave}, \texttt{Eva}\}$ is a winning committee with $\text{score}_{\text{AV}}(B) = 2 + 2 + 2 + 0 + 1 = 7$. As

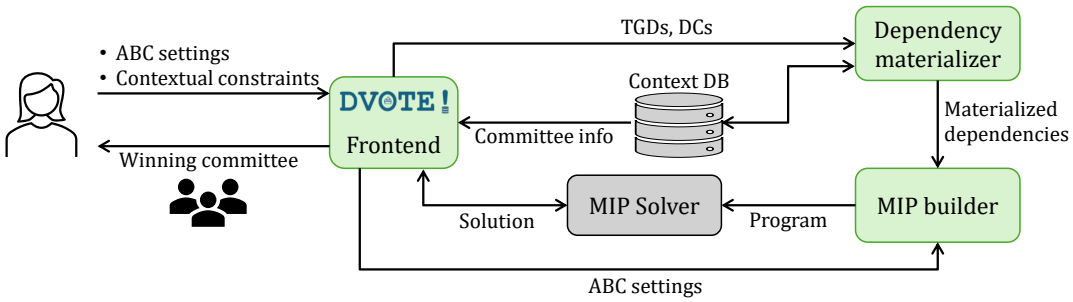

Figure 2: DVote implementation architecture. See Figure 3 for the UI of the frontend component.

another example, consider the following TGD:

$$\forall t\big[\mathsf{Topic}(t) \rightarrow \exists c, p[\mathsf{Author}(c,p) \wedge \mathsf{Pub}(p,t) \wedge \textsc{Com}(c)]\big]$$

This TGD states that the committee has at least one member with a publication on each topic. One can also include a specific TGD stating that at least one committee member should have both ML and PL publications (say, since the blend is central to the conference):

$$\textbf{true} \rightarrow \exists c, p, p'\big[\mathsf{Author}(c,p) \wedge \mathsf{Author}(c,p') \wedge \mathsf{Pub}(p,\mathtt{ML}) \wedge \mathsf{Pub}(p',\mathtt{PL}) \wedge \textsc{Com}(c)\big]$$

There is now one winning committee, {Ann, Cale, Dave}, since the three members cover all topics, and Ann has both ML and PL publications (namely p1 and p2), and it maximizes the score among all legal committees. As a last example, the rule

$$\forall c_1, c_2\big[\mathsf{Supervise}(c_1,c_2) \wedge \textsc{Com}(c_1) \wedge \textsc{Com}(c_2) \rightarrow \exists p[\mathsf{Author}(c_1,p) \wedge \mathsf{Pub}(p,\mathtt{ML})]\big]$$

states that we allow for supervision between committee members only if the advisor has an ML publication.

3 SYSTEM IMPLEMENTATION

DVote is a tool that encapsulates our framework for constrained ABC voting with a context database. It enables the user to choose the ABC settings, formulate contextual constraints (TGDs and DCs), find a winning committee, and analyze the winning committee. The main system components and design are depicted in Figure 2. The system deploys two external components: the database and the solver. The database contains the ABC information (candidate set C, voter set V, and approval profile A), as well as the relations of the context database D. The system is implemented in Python3 and we use SQLite3 as the database engine. For MIP solving, we use Google OR-Tools that wraps Gurobi 12.0.0.

Frontend. The frontend (shown in Figure 3) is implemented with Streamlit. It enables the user to choose different databases and ABC settings (①) such as committee size k and the scoring rule f. In the advanced settings, she can define the candidate set C and voter set V (④). She can then easily formulate any combination Γ of TGD and DC constraints, as defined in Section 2, using a graphical and interactive layout of the constraints (② and ③).

The graphical interface drastically facilitates the formulation of constraints compared to, say, writing SQL code to identify the conflicts in DCs or writing Datalog rules to represent the TGDs. We expect the user to know the schema of the context database and, as background, the general first-order structure of DCs and TGDs (as guided by DVote). With these, the user can phrase the constraints using selections from lists (e.g., relation/attribute names) and very few, if any, keyboard characters.

Once all is set, the user can invoke the voting algorithm to find a winning committee (⑤). When the backend is done processing, the results of the solver are sent back to the user interface, which parses it and queries the database D for further information about the members of the winning committee. It then presents the user with the resulting committee and further execution details (⑥).

Backend components. Within the backend, the *dependency materializer* gets as input the set Γ of contextual constraints, parses it, generates SQL queries, and executes them in the database. The results of these queries are components that can be turned into individual MIP constraints. (For example, a single DC, stating that no two committee members belong to the same party, materializes into pairs of candidates from the same party.) Using the ABC settings from the frontend and the output produced by the dependency materializer, the *MIP builder* constructs the MIP objective and constraints to be sent to the external MIP solver, while deploying optimizations to minimize the MIP. The full details of the optimized construction of the MIP, as well as runtime experiments, can be found in the framework paper [12, Section 5].

4 DEMONSTRATION

In the demonstration, we guide the audience in interaction with DVote, presenting a walk-through of how to find a winning committee on different databases, under different scoring rules and contextual constraints. Specifically, we show how to add increasingly complex combinations of contextual constraints to the committee selection. We view the results of each setting, detect deficiencies in the resulting committees, and add corresponding contextual constraints (i.e. database dependencies) to resolve them.

We use several databases from different domains, including political elections,[1] hotels,[2] and movies,[3] which we also use here for illustration. These databases are described in detail in the archive paper [12, Section 6.1]. Each database is associated with a committee election task such as a movie theater that selects a set of

[1] preflib.github.io/PrefLib-Jekyll/dataset/00008, accessed July 12, 2025
[2] preflib.github.io/PrefLib-Jekyll/dataset/00040, accessed July 12, 2025
[3] www.kaggle.com/datasets/rounakbanik/the-movies-dataset, accessed July 12, 2025

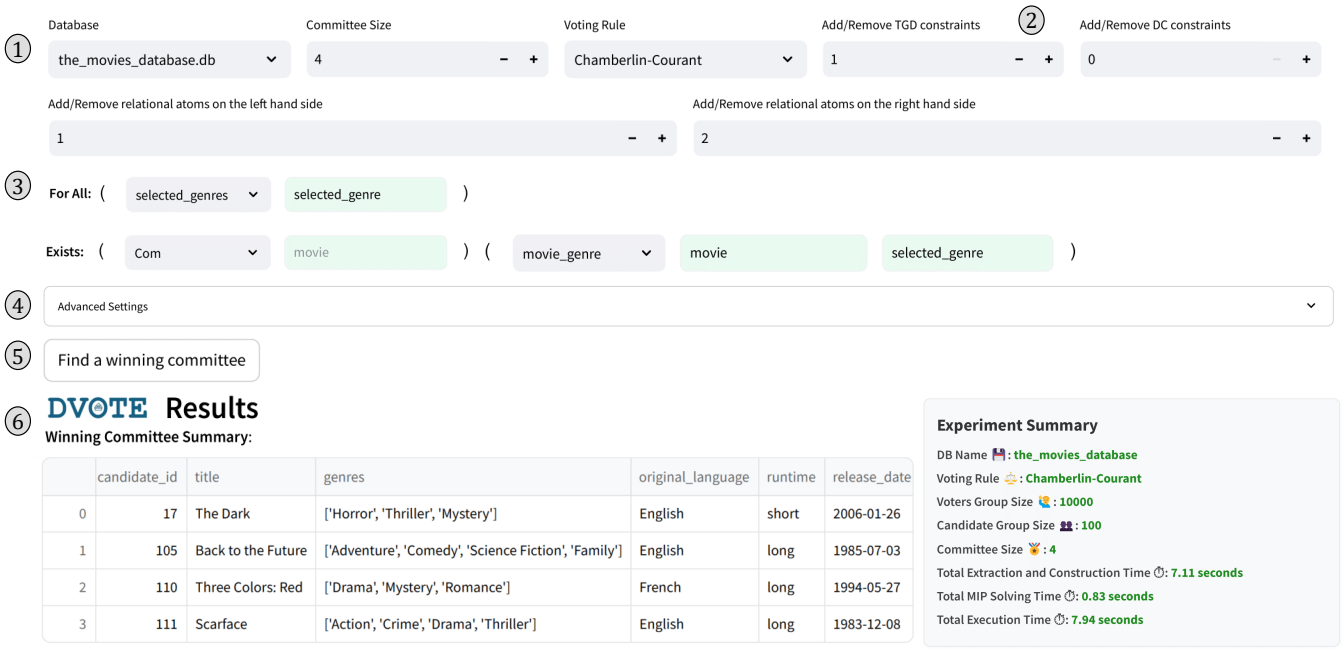

Figure 3: DVote user interface.

movies for a weekend screening, a company pursuing special price contracts for its employee travels, and a city electing a council.

No Constraints. We begin with simple committee elections to familiarize ourselves with ABC voting. Specifically, we use DVote to find winning committees with varying sizes and ABC scoring rules (such as AV, CC, and PAV). For the different ABC settings, we inspect the unconstrained winning committee and the execution time. (Recall that unconstrained ABC is already NP-hard for common rules, with the exception of AV [2, 10].)

Basic Constraints. After observing the unconstrained winning committees, we add simple DCs and TGDs to constrain the considered committees to satisfy different needs. For example, the movie committee misses a movie with an important popular genre such as Comedy, or lacks diversity in language, so we resolve these using TGDs. In the hotel committee, there are two hotels with the same location and price range, so we add a DC to avoid it. In the political election, we add a TGD to demand representation for all wards in the elected council. This scenario demonstrates the usefulness and ease of constraining committee elections in DVote.

Advanced Constraints. Next, we delve into the different use cases and illustrate complex combinations of contextual constraints to capture more subtle organizational needs. As an example, "for every selected high-priced hotel, there is also a cheap alternative at the same location" (using a TGD that has the relation symbol Com in both the premise and conclusion of the logical implication). For the council election, we enforce that there are no three members from the same party (hence, avoiding over-representation of a party). This scenario demonstrates the expressiveness of DVote and its ability to capture highly specialized needs.

ACKNOWLEDGMENTS

The authors are extremely grateful to Phokion Kolaitis for fruitful and insightful discussions on the model and implementation underlying the demonstration.

REFERENCES

[1] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh. 2015. Justified Representation in Approval-Based Committee Voting. In *AAAI*. AAAI Press, 784–790.
[2] Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Simon Mackenzie, Nicholas Mattei, and Toby Walsh. 2015. Computational Aspects of Multi-Winner Approval Voting. In *AAMAS*. ACM, 107–115.
[3] Catriel Beeri and Moshe Y. Vardi. 1984. A Proof Procedure for Data Dependencies. *J. ACM* 31, 4 (1984), 718–741.
[4] Leopoldo E. Bertossi and Jan Chomicki. 2003. Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*. Springer, 43–83.
[5] Robert Bredereck, Piotr Faliszewski, Ayumi Igarashi, Martin Lackner, and Piotr Skowron. 2018. Multiwinner Elections With Diversity Constraints. In *AAAI*. AAAI Press, 933–940.
[6] Benny Kimelfeld, Phokion G. Kolaitis, and Julia Stoyanovich. 2018. Computational Social Choice Meets Databases. In *IJCAI*. ijcai.org, 317–323.
[7] Benny Kimelfeld, Phokion G. Kolaitis, and Muhammad Tibi. 2019. Query Evaluation in Election Databases. In *PODS*. ACM, 32–46.
[8] Martin Lackner and Piotr Skowron. 2018. Approval-Based Multi-Winner Rules and Strategic Voting. In *IJCAI*. ijcai.org, 340–346.
[9] Martin Lackner and Piotr Skowron. 2021. Consistent approval-based multi-winner rules. *J. Econ. Theory* 192 (2021), 105173. https://doi.org/10.1016/J.JET.2020.105173
[10] Piotr Skowron, Piotr Faliszewski, and Jérôme Lang. 2016. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artif. Intell.* 241 (2016), 191–216. https://doi.org/10.1016/J.ARTINT.2016.09.003
[11] Yongjie Yang and Jianxin Wang. 2018. Multiwinner Voting with Restricted Admissible Sets: Complexity and Strategyproofness. In *IJCAI*. ijcai.org, 576–582.
[12] Roi Yona and Benny Kimelfeld. 2025. Using Database Dependencies to Constrain Approval-Based Committee Voting in the Presence of Context. arXiv:2501.16574 [cs.DB] https://arxiv.org/abs/2501.16574