OPRA: An Open-Source Online Preference Reporting and Aggregation System

Yiwei Chen,[1] Jingwen Qian, [2] Junming Wang, [3] Lirong Xia, [2] Gavriel Zahavi
[1] New York University
[2] Rensselaer Polytechnic Institute
[3] Stanford University
reedyka3@gmail.com, qianj2@rpi.edu, jumingw@cs.stanford.edu, xial@cs.rpi.edu, gavriel.zahavi@gmail.com

Abstract

We introduce the Online Preference Reporting and Aggregation (OPRA) system, an open-source online system that aims at providing support for group decision-making. We illustrate OPRA's distinctive features: UI for reporting rankings with ties, comprehensive analytics of preferences, and group decision-making in combinatorial domains. We also discuss our work in an automatic mentor matching system. We hope that the open-source nature of OPRA will foster development of computerized group decision support systems.

Introduction

The field of *social choice*, sometimes referred to as *group decision-making* or *collective decision-making*, aims at designing mechanisms to help people make a joint decision, despite that they may have conflicting preferences. Typical examples of social choice are (1) **voting**, ranging from high-stakes (e.g. presidential elections) to low-stakes (e.g. deciding a restaurant for dinner) voting scenarios; and (2) **resource allocation**, ranging from high-stakes (e.g. allocating resources among countries) to low-stakes (allocating course projects to students).

In this paper, we introduce a system for online group decision-making: the Online Preference Reporting and Aggregation (OPRA) system,[1] which has the following distinctive features:

User interfaces for reporting rankings with ties. OPRA provides 5 intuitive interfaces for users to rank the alternatives with ties. In particular, the *one-column* and *two-column* UIs allow users to report rankings with ties via drag-and-drop operations.

Comprehensive analytics of preferences. In addition to standard information and statistics about voting outcomes, OPRA also computes and displays other consensus metrics, such as clusters of preferences based on learning mixture of Plackett-Luce models (Zhao, Piech, and Xia 2016), and margin of victory (Xia 2012).

Group decision-making in combinatorial domains. OPRA supports reporting of Conditional Preference Networks (CP-nets) (Boutilier et al. 2004) for multi-issue voting (Lang and Xia 2016) and multi-type resource allocation (Mackin and Xia 2016).

Matching OPRA includes a system for automatic assignment of student mentors to courses based on their preferences and qualifications.

Open-source. OPRA is open-source, allowing easy modification and customization for deployment, and easy adaptation of its components to other systems.

Voting UIs

OPRA includes the following five UIs for users to report weak orders over alternatives. (1) **One-Column (Figure 1).** OPRA's one-column UI is an extension of the widely-used JQuery sortable class. Users perform drag-and-drop operations to achieve the rank order they desire (Li et al. 2019). This UI supports ties among alternatives.(2) **Two-Column.** The two-column UI is a variation of the one-column UI. The left column is the same as one-column, and the right column allows clicking operation to rank. the alternatives that remain in the right column are submitted as unranked. (3) **Sliders.** each alternative is ranked from 0 to 100 using a slider. (4) **Star Rating.** each alternative is ranked from 0 to 10 using a star rating UI. (5) **Yes/No.** each alternative is associated with a checkbox allowing submission of approval vote.

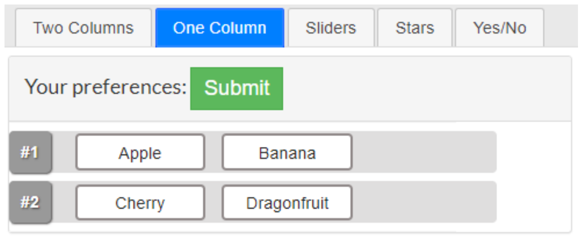


Figure 1: OPRA's one column voting UI.

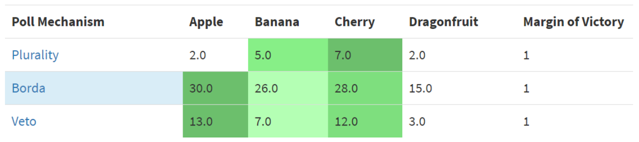
Comprehensive Analytics of Preferences

OPRA computes and shows the results of some voting rules, preference learning, and margin of victory.

Commonly-studied voting rules. OPRA implements commonly-studied voting rules, including Plurality, Borda,

[1]site: opra.io; source code: github.com/PrefPy/opra

k-Approval, Single Transferable Vote (STV), and others. For a poll, each voting rule is applied. The result is displayed in a table, and the winner can be seen under different voting rules (Figure 2 below shows some rows from an example table). While most voting rules are implemented using standard algorithms based on their definitions, some voting rules' implementation uses state-of-the-art algorithms proposed in recent literature. For example, implementation of STV and Ranked pairs computes all parallel universe tie-breaking (PUT) winners (Conitzer, Rognlie, and Xia 2009) using a DFS-based algorithm in (Wang et al. 2019).



Poll Mechanism	Apple	Banana	Cherry	Dragonfruit	Margin of Victory
Plurality	2.0	5.0	7.0	2.0	1
Borda	30.0	26.0	28.0	15.0	1
Veto	13.0	7.0	12.0	3.0	1

Figure 2: An example of poll result table on OPRA. Here, Cherry wins under Plurality, but Apple wins under Borda and Veto.

Margin of victory. OPRA computes and shows the *margin of victory* (Magrino et al. 2011; Cary 2011; Xia 2012). Margin of victory measures the robustness of the voting outcome, and also plays an important role in post-election audits. The higher the margin of victory is, the more robust the outcome is.

Preference learning. OPRA implements the EGMM method (Zhao, Piech, and Xia 2016) of learning mixtures of Plackett-Luce model. Learned mixtures offer a high-level overview of groups and strength of preferences based on users' ordinal preferences and can be used to make useful decisions such as group activity selection (Darmann et al. 2012).

Group Decision-Making in Combinatorial Domains

In a combinatorial domain, there are exponentially many alternatives, each of which is characterized by its values on p variables. For example, in combinatorial voting (Lang and Xia 2016), voters vote to decide approval/non-approval of p issues. In multi-type resource allocation (Mackin and Xia 2016), there are p types of items to be allocated to agents according to their preferences.

The challenges in group decision-making in combinatorial domains are: **First**, there are often exponentially many alternatives/bundles of items, which makes it hard for users to report their preferences. **Second**, it is unclear what mechanisms should be applied when users use a compact language, e.g. CP-nets as we will recall shortly, to represent their preferences. See (Lang and Xia 2016) for more discussions.

To address the first challenge, OPRA allows users submit CP-nets (Boutilier et al. 2004). To address the second challenge, OPRA supports *sequential voting* (Lang 2007) on multiple issues and *sequential resource allocation* of multiple types of items, via the "Multi-Poll" section.

To execute sequential voting, OPRA allows users to express preferences either using a CP-net, indicating their conditional preferences, or by waiting until previous sub-polls finish, then submit their preferences over the current issue.

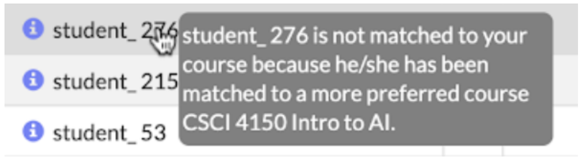
For sequential resource allocation, OPRA allows users to use *serial dictatorships* to allocate indivisible items, when the number of users is the same as the number of items.

Matching

OPRA offers an online Mentor application tool to automatically assign student mentors to courses looking for mentors. The matching is based on the kinds of mentors a course is looking for, and the courses desired for mentoring by a student. For each course, instructors and administrators are able to modify the weights of its features (e.g. how important are GPA, mentor experience, etc.), while from the student's side, they can submit their application form and their rankings of courses. The personal features and course preferences will be taken as their input of the matching algorithm. Administrators can also modify weights, add, remove, or fix students through the ranking UI, and perform real-time re-matching.

Matching algorithm. For a given course, we calculate a student's score for that course as the dot product between the course's weight vector and the student's feature vector. this is done for each student, the resulting scores giving a preference ranking for this course over all the students. we then use these courses' rankings over students and students' (partial) rankings over courses to perform a course-proposing Gale-Shapley stable matching algorithm (Gale and Shapley 1962) (with courses also having a max capacity). This process can be re-run by administrators at any time.

Explainable AI. The system also supports some Explainable AI functionality (Ribera and Lapedriza 2019). From their page, administrators can view reasons why a given student was a assigned to a class or not. For example, a student wasn't assigned to the class because they were assigned to a higher-ranked class instead, as seen in Figure 3.



student_276
student_215
student_53

student_276 is not matched to your course because he/she has been matched to a more preferred course CSCI 4150 Intro to AI.

Figure 3: An example of Explainable AI on OPRA.

Future Work

- **Usability:** Improve human computer interaction, such as offering initial rankings to users without introducing bias.
- **Explainability:** Add more comprehensive analytics and statistics for the purpose of better explanations.
- **Algorithms:** Improve algorithms for polls to aid in group activity selection and group decision making, as well as improve matching algorithms to handle min-max bounds to course capacity (Nasre and Nimbhorkar 2017).

Acknowledgments

This work is supported by NSF #1453542, NSF #1716333, and ONR #N00014-17-1-2621.

References

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus statements. *Journal of Artificial Intelligence Research* 21: 135–191.

Cary, D. 2011. Estimating the Margin of Victory for Instant-Runoff Voting. In *Proceedings of 2011 EVT/WOTE Conference*.

Conitzer, V.; Rognlie, M.; and Xia, L. 2009. Preference Functions That Score Rankings and Maximum Likelihood Estimation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 109–115. Pasadena, CA, USA.

Darmann, A.; Elkind, E.; Kurz, S.; Lang, J.; Schauer, J.; and Woeginger, G. 2012. Group activity selection problem. In *Proceedings of the 8th international conference on Internet and Network Economics*, 156–169.

Gale, D.; and Shapley, L. S. 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly* 69(1): 9–15.

Lang, J. 2007. Vote and Aggregation in Combinatorial Domains with Structured Preferences. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, 1366–1371. Hyderabad, India.

Lang, J.; and Xia, L. 2016. Voting in Combinatorial Domains. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A., eds., *Handbook of Computational Social Choice*, chapter 9. Cambridge University Press.

Li, H.; Sikdar, S.; Vaish, R.; Wang, J.; Xia, L.; and Ye, C. 2019. Minimizing time-to-rank: a learning and recommendation approach. *arXiv preprint arXiv:1905.11984* .

Mackin, E.; and Xia, L. 2016. Allocating Indivisible Items in Categorized Domains. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 359–365.

Magrino, T. R.; Rivest, R. L.; Shen, E.; and Wagner, D. 2011. Computing the Margin of Victory in IRV Elections. In *Proceedings of 2011 EVT/WOTE Conference*.

Nasre, M.; and Nimbhorkar, P. 2017. Popular Matching with Lower Quotas. *arXiv preprint arXiv:1704.07546* .

Ribera, M.; and Lapedriza, A. 2019. Can we do better explanations? A proposal of user-centered explainable AI. In *IUI Workshops*.

Wang, J.; Sikdar, S.; Shepherd, T.; Zhao, Z.; Jiang, C.; and Xia, L. 2019. Practical Algorithms for STV and Ranked Pairs with Parallel Universes Tiebreaking. In *Proceedings of AAAI*.

Xia, L. 2012. Computing The Margin of Victory for Various Voting Rules. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 982–999. Valencia, Spain.

Zhao, Z.; Piech, P.; and Xia, L. 2016. Learning Mixtures of Plackett-Luce Models. In *Proceedings of the 33rd International Conference on Machine Learning*.