EZ-PC: Program Committee Selection Made Easy

Vyas Sekar
Carnegie Mellon University
vsekar@andrew.cmu.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed. The author takes full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

Selecting a technical program committee (PC) for a conference or a workshop can be an intimidating and time consuming process. PC selection needs to balance several potential considerations; e.g., industry vs. academic participation, inclusion of under-represented communities, ensuring coverage over topic areas, among others. This paper presents the design of an open-source tool called EZ-PC which formulates these considerations as a simple constraint satisfaction problem to help PC chairs systematize this selection process. We report on some of the features we have incorporated and our experience in using the tool.

1. INTRODUCTION

A necessary and critical step in putting together a technical program for a conference is to select a high quality *program committee (PC)*. The goal of the PC is to review submissions, provide expert reviews, and ultimately converge on the best possible technical program for the given conference or journal.

Selecting a PC entails balancing a wide range of requirements and practical considerations, including:

- *Topic coverage:* Given the growing breadth of many technical fields and the emergence of sub-areas of interest, most technical conferences span a broad range of topics. The technical program chairs need to ensure that there is sufficient representation from different areas. A lack of expertise in a particular sub-area affects the ability to judge the novelty and correctness of the submissions and also means that the chairs may have to seek several out-of-band reviews. This may further impact the process, as these experts may not have a full view of the overall submission pool for calibration.
- *Representation from diverse groups:* PC chairs often strive to encourage participation from diverse and under-represented communities. This is especially important to broaden participation and provide valuable experience to members of the community in organizing top-quality conferences. These may include both geographical considerations (e.g., avoid a US-centric PC) and seniority factors (e.g., ensure there is a good mix of senior, mid-career, and junior members of the community).
- *Avoiding over-representation from specific groups:* An equally important concern is over-representation from specific subgroups. For instance, it is useful to avoid having too many members from the same institution as it may make it hard to find expert reviews when considering institutional conflicts. Similarly, it may also be useful to ensure that specific sub areas (e.g., "hot" topics) are not over-represented to avoid inducing a systematic bias in the program composition.
- *PC Size:* Depending on the expected number of submissions and the intended workload, one also needs to keep the PC size manageable. A small PC may make for more engaged discussions and better calibration of the PC members to the rest of the submission pool, but raises the risk of PC fatigue and overload. A large PC on the other hand may help distribute the load and may make it easier to meet some of the coverage constraints described above, but incurs the risk of weaker discussions.
- *Iterative process:* A practical logistical constraint is in the availability of candidate PC members and their responses. Even if we optimistically assume a response rate of $\approx 75\%$, this means that the PC selection process will proceed in multiple rounds as the chairs learn of candidates' availability. Thus, we also need to ensure that these are satisfied over this iterative process.

Based on anecdotal evidence, we believe that the process that PC chairs attempt to address these aforementioned considerations is largely manual. While this is not entirely unreasonable to handle manually (e.g., even the largest PCs for single track conferences we know of have ≈ 70 members), it is quite tedious and may invariably introduce one or more potential blind spots in terms of the coverage and representation concerns.

To simplify the PC chairs job of balancing these considerations, we developed a simple open-source toolkit called **EZ-PC**[1] to systematically formulate these factors and automate the PC selection process [1]. In a nutshell, EZ-PC expresses these constraints as a simple *integer linear program (ILP)* and uses off-the-shelf solvers to find feasible solutions. We do not intend to claim that EZ-PC's design is novel or technically interesting; it is simply a useful tool to codify the typical considerations that PC chairs face.

In the rest of this short editorial, we describe the design of EZ-PC and our preliminary experience in using it to automate the PC selection process.

2. EZ-PC FORMULATION

In this section, we formally specify the various considerations in PC selection and describe the Integer Linear Program (ILP) formulation we use in EZ-PC.

Inputs: We begin by describing the inputs into EZ-PC and then describe how use these inputs to generate the ILP formulation.

- *Features:* Recall that there were different considerations that need to be covered; e.g., Areas of interest, Geographical constraints, Underrepresented communities, Seniority etc. We model each of these considerations as a *binary feature*. Let $\mathcal{F}$ denote the set of all binary features and let $f \in \mathcal{F}$ refer to a specific feature from this set. Note that these features need not be orthog-

[1]Wordplay intended!

Minimize: $PCSize$, subject to

$$PCSize = \sum_{c \in \mathcal{C}} select_c \quad (1)$$

$$\mathbf{MinPCSize} \leq PCSize \leq \mathbf{MaxPCSize} \quad (2)$$

$$\forall f \in \mathcal{F} : \sum_{c \in \mathcal{C}} \mathbf{I}_{c,f} \times select_c \geq \mathbf{MinCoverage}_f \quad (3)$$

$$\forall f \in \mathcal{F} : \sum_{c \in \mathcal{C}} \mathbf{I}_{c,f} \times select_c \leq \mathbf{MaxCoverage}_f \quad (4)$$

$$\forall g \in \mathcal{G} : \sum_{c \in g} select_c \leq \mathbf{GroupUpper}_g \quad (5)$$

$$\forall c \in \mathbf{Responses} : select_c = \mathbf{Availability}_c \quad (6)$$

$$\forall c : select_c \in \{0, 1\} \quad (7)$$

Figure 1: ILP for the PC selection problem

onal and in fact will not be so by design; e.g., the geographical and topic characteristics are not orthogonal dimensions.

- *Candidates:* We assume that the PC chairs have prepared (manually or otherwise), a set of candidate PC members $\mathcal{C}$. Let $c \in \mathcal{C}$ refer to a specific PC candidate. Each candidate c's attributes with respect to the features of interest needs to be populated; we use the binary indicators $\mathbf{I}_{c,f}$ to denote if the candidate c has the binary feature f "on". For instance, if the feature is a specific sub-area (e.g., TCP) and the candidate is an expert in this topic then this indicator will be set to 1. We can similarly model geographical properties (e.g., Asia vs. EU vs. US) and indicating whether the candidate is from a specific under-represented group.
- *Coverage requirements:* As discussed earlier, for each feature of interest, we have two kinds of constraints on the coverage. First, for each type of feature we have a *minimum coverage* level $\mathbf{MinCoverage}_f$ denoting the minimum number of PC members who satisfy this particular feature; e.g., at least 5 junior members and at least 4 people from region X. Second, we also want to avoid over-representation from specific groups. To this end, we introduce an optional upper bound on the coverage as well denoted as $\mathbf{MaxCoverage}_f$ which denotes the maximum number of PC members satisfying this feature.
- *Group constraints:* Some organizations (e.g., large research labs or large research universities) may invariably have a large representation on the PC. To avoid this, we introduce the notion of group constraints that allow us to specify an upper bound on the number of members from a specific group. We have a set of user-defined groups $\mathcal{G}$. For each group, $g \in \mathcal{G}$, we have an upper bound on the number of candidates from that group that can simultaneously be on the PC $\mathbf{GroupUpper}_g$.[2]
- *PC size:* Recall that to balance the reviewing workload, we also want to have a minimum PC size depending on the number of expected submissions and the expected number of reviews per paper. Let $\mathbf{MinPCSize}$ denote this minimum PC size. Similarly, we want to make sure the PC is not too large; let $\mathbf{MaxPCSize}$ denote the maximum possible PC size we want to have.

[2]We could have also modeled these as features and used the $\mathbf{MaxCoverage}$ constraints to capture these group constraints. Since these types of groups were limited in number, it was more convenient to express these separately rather than a new feature per-group.

- *Availability and prior selection:* As chairs send invitations and candidates indicate their availability or unavailability, we may need to iteratively re-run the EZ-PC tool and update the selection based on these candidate constraints. To this end, we also have to maintain an updated record of the availability of the different candidates as they respond. Let $\mathbf{Responses}$ denote the set of candidates who have already provided responses and let $\mathbf{Availability}_c$ denote the expressed availability (or lack) as a binary indicator. We use these indicators to also denote candidates who have already been selected and marked as available in previous rounds to ensure that they continue to be selected in subsequent rounds.

ILP Formulation: Given these inputs, next we describe how we formulate PC selection using an ILP as shown in Figure 1.

We introduce the *decision variables* $select_c \in \{0, 1\}$ to capture the decision process if a particular candidate c has been selected to serve on the PC. Our goal is to set some of these to 1 to select the actual PC to meet the constraints described earlier. We use a simple objective function, which is to minimize the total PC size subject to several constraints modeling the coverage, group, and availability considerations. This objective ensures that the solver program prefers a smaller PC subject to the other constraints, including the minimum PC size.

Our constraints naturally map to the considerations we raised earlier. Eq (1) models the PC size as the sum of the $select$ decision variables. Eq (2) models the upper and lower bounds on our PC size. ($PCSize$ is a convenient temporary variable for clarity; we can write the entire formulation in terms of the $select$ decision variables alone.) Eq (3) and (4) model the coverage requirements per feature of interest in terms of the indicators and the decision variables. (Note that the $\mathbf{I}$ values in the equations are constants rather than variables, which makes our problem a simple integer linear program.) Eq (5) ensures that for each group, we have a limit on the number of simultaneous candidates chosen from that group. Finally, to capture the iterative process, we introduce the availability constraints for the candidates who have previously responded (i.e., in $\mathbf{Responses}$) in Eq (6). This ensures that the subsequent runs of the ILP will honor the previous selection and unavailability rather than creating a solution from scratch that may violate some of these constraints. In general, in the first round of invitations, the set of $\mathbf{Responses}$ will be empty and these constraints can be ignored; here, we show the general formulation. Finally, we have the binary constraints on each decision variable.

3. IMPLEMENTATION AND WORKFLOW

In this section, we briefly describe the implementation of the EZ-PC tool. EZ-PC is currently a simple command line tool written in `Perl` that takes as input a few text files (described below) and uses `glpsol` [2] as the underlying ILP solver. EZ-PC has very few dependencies with external libraries, and the only requirement is to have a working version of `glpsol`. We used Perl v5.16.3 built for darwin-thread-multi-2level and glpsol v4.48 installed through `MacPorts` [3].

To use EZ-PC, the PC chairs need to populate four key text files with the following formats:

1. *Candidate Feature Values:* This is a simple CSV (comma separated values) file. The first column specifies the candidate name, and the remaining columns with a binary indicator (1 or 0) indicating whether the candidate satisfies the feature. (The first row has the feature names.) For instance,

```
Name,Area1,Area2,Area3
```

```
Alice,1,0,1
Bob,0,1,1
Eve,0,0,1
..
```

specifies that Alice "covers" Area1 and Area3, while Bob covers Area2 and Area3, while Eve only covers Area3.

2. *Feature Constraints:* This is a CSV file specifying features and their **MinCoverage** and **MaxCoverage** constraints, one per line, with the first column being the feature name and the remaining being the min-max values. If there are features without any constraints, these need not appear in the file. A simple way to avoid the **MaxCoverage** is to set it to the **MaxPCSize**. Note that the names of the features in this file should match the first row of the Candidates file above. For instance, to specify a minimum of three members covering Area1 and Area2, we would have:

   ```
   Area1,3,50
   ```

3. *Availability Constraints:* A two-column CSV marking a 0 for a candidate who has already declined and 1 for a candidate who has already accepted (i.e., selected in a previous round of EZ-PC selection). In the first round of PC invitations, this file will typically be empty. For instance, to specify that Alice has already agreed and Bob has already declined, we would have:

   ```
   Alice,1
   Bob,0
   ```

4. *Group Constraints:* A space-separated file, with the first column giving a name for the group, the second column giving the comma-separated list of group members and the third column giving the **GroupUpper** value for this group. Each group can be arbitrarily large. (Note that groups may overlap.) For instance, if Alice, Bob, and Eve are from the same organization XYZ and we do not want more than 2 of them to be simultaneously selected, we specify

   ```
   XYZ Alice,Bob,Eve 2
   ```

EZ-PC consists of two basic `Perl` scripts: (1) to generate the ILP formulation and the solution and (2) to parse the solution output by the ILP to extract the list of selected candidates from the solution output.

Scalability: We have encountered almost no scalability problems in using EZ-PC so far. For a set of 87 candidate PC members with 23 features, and having a minimum of 45 PC members to be selected, the run time was less than 0.5 seconds on a Macbook Pro with a 2.4 GHz Intel Core i5 processor and 8GB of RAM. We have tried with other configurations and the run time was consistently less than 1 second, so we anticipate that scalability will not be a problem for the typical PC size and types of features we expect.

4. CONCLUDING REMARKS

EZ-PC certainly made our life easier to ensure various types of coverage constraints and balancing requirements. That said, EZ-PC is very much an "alpha" stage tool with several quirks reflecting the need to get working code as our needs demanded. Our wishlist for features is quite numerous. First on the list is the need suitable data preprocessing steps (e.g., conversion from spreadsheet into EZ-PC-compatible text files that avoid special characters in candidate and feature names). Second, a nice feature to add would be to inform the chairs why and how the problem might be infeasible when it is. For instance, when we have PC members who cover a lot of areas, then the **MaxCoverage** constraint often gets violated and in this case we had to manually increase the value to accommodate this. Having some way to the PC chairs understand the ILP output and infeasibility scenarios would be a useful addition. Third, EZ-PC is currently two command line programs and we are planning to interface it with better web-based or graphical interfaces to make it more broadly usable. Finally, the process of generating EZ-PC inputs is manual; one way to automate it is to crawl public resources (e.g., Google Scholar, DBLP, and recent conference proceedings) to identify candidates and their areas of expertise rather than have the PC chairs input these manually.

The latest version of EZ-PC can be downloaded at: `http://users.ece.cmu.edu/~vsekar/ezpc.html`

Acknowledgments

The author would like to thank Dejan Kostic and the ACM CoNext steering committee for their inputs that motivated the need for and informed the design of EZ-PC. EZ-PC also benefited from early conversations with Petros Maniatis.

Please send comments or suggestions on improving the EZ-PC tool to `vsekar@andrew.cmu.edu`.

5. REFERENCES

[1] EZ-PC: Download and Documentation. `http://users.ece.cmu.edu/~vsekar/ezpc.html`.

[2] GLPK: GNU Linear Programming Kit. `https://www.gnu.org/software/glpk/`.

[3] The MacPorts Project. `https://www.macports.org/`.