EZ-PC: Program Committee Selection Made Easy

Vyas Sekar, Carnegie Mellon University

ABSTRACT

Selecting a technical program committee (PC) for a conference or a workshop can be a somewhat 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. The goal of this short paper is to document our experience of building an open source tool called EZ-PC. In a nutshell, EZ-PC helps formulate some of 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 experiences in building and using the tool.

1 Introduction

Perhaps the most important and critical step that program co-chairs for a conference do is putting together a technical program is selecting a high quality *program committee (PC)*. The goal of the PC as many of us know 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 constraints, including:

- *Area coverage:* Given the growing breadth of many of our technical fields and the emergence of sub-areas of interest, most technical conferences sponsored by ACM/IEEE have a wide diversity of topics. A critical concern for the technical program chairs is to ensure that there is sufficient representation from different areas in which the community works on. A lack of expertise in a particular sub-area is especially worrisome in terms of the ability to judge the novelty and correctness of the submissions. Furthermore, this also means that the chairs may have to seek several more out-of-band reviews from experts, who may not have a panoptic view of the overall quality of the submissions in rating the given submission.
- *Ensuring representation from diverse groups:* PC chairs on the advice of the steering committee often strive to encourage participation from diverse and under-represented communities. This is especially important to broaden the participation and give such communities the exposure to gaining valuable experience in organizing top-quality conferences. For instance, these representation considerations may include geographical (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 risk 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 risks weaker discussions.

Based on our conversations with former PC chairs of various conferences, PC chairs attempt to manually address these aforementioned considerations. While the scale 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 cumbersome and invariably introduces one or more potential blind spots in terms of the coverage and representation concerns.

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

In the rest of this short paper, we describe the design of EZ-PC and our experiences in using it to automate the PC selection process. We also describe some of iterative refinement of the tool that we needed to introduce during the process. For instance, a practical logistical constraint is in the

[1] wordplay intended!

availability of potential PC members and their responses indicating the availability. Even optimistically assuming a response rate of $\approx 75\%$, this means that the PC selection process will invariably proceed in multiple rounds as the chairs learn of the availability of the candidates. Thus, not only do we need to capture the above considerations, but also ensure that these are satisfied over this iterative process, so we introduced new features to the tool to express these availability constraints as well.

2 EZ-PC Formulation

In this section, we describe how we formally specify the various considerations in PC selection outlined in the previous section. To this end, we describe the specific Integer Linear Program (ILP) that EZ-PC uses.

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

- *Features of Interest:* 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 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 independent and/or orthogonal and in fact will not be so by design; i.e., the geographical and area characteristics of a given candidate will inherently have some overlap.
- *Candidate List:* 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. Now, each c's information with regards to the various features of interest needs to be populated; we use the binary indicator constants $\mathbf{I}_{c,f}$ to indicate 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 constant will be set to 1. Similarly, we do the same for geographical properties (e.g., Asia vs. EU vs. US) and whether the candidate is from a specific under-represented group.

  We introduce the *decision variables* $select_c \in \{0, 1\}$ to capture the decision process if a particular candidate c needs to be selected.
- *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., we may want at least 5 junior members of the community and at least 4 people from Asia on the PC. Second, we also noted that we wanted 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:* One observation we made as we formulated the problem was that some organizations (e.g., large research labs or large research universities) may invariably have a large representation on the PC. To this end, 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 groups $g \in \mathcal{G}$ and for each group, we have an upper bound on the number of candidates from that group that can simultaneously be on the PC $\mathbf{GroupUpper}_g$. In some sense, we could have also modeled these as features and used the $\mathbf{MaxCoverage}$ constraints to capture these group constraints as well. Since these types of groups were limited in number, it was more convenient to express these separately rather than as a separate feature per-group.
- *Minimum 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 $MinPCSize$ denote this minimum PC size. Similarly, we want to make sure the PC is not too large; let $MaxPCSize$ denote the maximum possible PC size we want to have.
- *Candidate constraints:* As we roll out invitations and candidates indicate their availability or unavailability, we may need to iteratively refine the PC selection process 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 $Availability_c$ denote the expressed availability (or lack) as a binary indicator.

Minimize: $PCSize$, subject to

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

$$MinPCSize \leq PCSize \leq MaxPCSize \quad (2)$$

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

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

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

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

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

Figure 1: ILP for the PC selection problem

ILP Formulation: Given these inputs, next we describe how we formulate PC selection as a simple ILP as shown in Figure 1. We introduce 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 was to ensure that the constraint 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 decision variables and Eq (2) models the upper and lower bounds on our PC size. (The $PCSize$ 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.) Then, Eq (5) ensures that for each of our groups, we have a cap 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 the individual decision variables.

3 Implementation and Workflow

In this section, we briefly describe the implementation of the EZ-PC toolkit. EZ-PC is very simple `Perl`-based toolkit that takes as input a few simple text files and uses `glpsol` [1] as the underlying ILP solver. EZ-PC has 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` [2].)

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

1. *Candidates' Features:* This is a simple CSV (comma separated values) file, with the number of columns equal to the number of features plus one column for the name. 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,[2]

```
Name,Area1,Area2,Area3
Alice,1,0,1
Bob,0,1,1
Eve,0,0,1
..
```

2. *Feature Constraints:* Another simple CSV file specifying the names of features and their $\mathbf{MinCoverage}$ and $\mathbf{MaxCoverage}$ 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 $\mathbf{MaxCoverage}$ 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
Area2,3,50
```

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. 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 $\mathbf{GroupUpper}$ value for this group. Each group can be arbitrarily large and there are no constraints that the groups may be overlapping. 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
```

There are two basic `Perl` files: one to generate the ILP formulation and the solution and the other 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/features for common networking conferences.

4 Conclusions

EZ-PC certainly made our life easier in terms of ensuring various types of coverage constraints and PC balancing re-

[2]There is a current quirk that names cannot have special characters in the formulation file.

quirements. EZ-PC is very much in an "alpha-minus" stage with little to no documentation and several quirks reflecting the need to get working code as our needs demanded. The latest version of EZ-PC can be downloaded at: `http://users.ece.cmu.edu/~vsekar/ezpc.html`

EZ-PC is far from perfect and our wishlist for features is quite numerous. First, we acknowledge that there are non-trivial quirks in data processing (e.g., conversion from spreadsheet into EZ-PC-compatible text files) that should be easy to address. Second, one thing EZ-PC does not do is tell us why/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 has no user interface whatsoever; it is simply two scripts running from the command line. While this has served our purpose, we acknowledge it might be useful to integrate EZ-PC into other programs; e.g., spreadsheets for PC selection. Finally, the process of generating EZ-PC inputs is manual; one way to automate it is to scrape public resources like Google scholar and/or DBLP and recent conference proceedings to identify candidates and their areas of expertise rather than have the PC chairs input these manually.

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] Gnu lp solver. .
[2] Gnu lp solver. .