A Web-Based Tabulation System

Benjamin Hebert

6.UAP Final Report

5/16/05
Abstract

I present a web-based system for tabulation American Parliamentary Debate Association (APDA) debate tournaments [1]. The system is designed for use by computer-literate, non-technical users. It provides tournament registration, tabulation, judge assignment, and room assignment features, in accordance with typical APDA practices [2]. The system is implemented on a web server owned by the APDA organization, and must maintain a high level of availability. The core algorithm for tabulation is the minimum-weight perfect matching algorithm [3], a well-known problem requiring polynomial time. The system uses an open-source implementation of that algorithm from the JICOS project [4].
Figures and Tables

Figure 1: System Layout ........................................................................................................... 7

Table 1: Tables in MySQL Database ....................................................................................... 9
1. Introduction

The goal of this project was to create a reliable system for tabulating tournaments. Using a “web-service” system design allows configuration of the software to be done once, on the server, rather than every time a new user wants to use the software. Moreover, it facilitates easy access and alteration of the program during tournaments, in the event of an error. Given the prevalence of reliable, high speed internet connections on college campuses, requiring internet access on the part of the user does not create a significant burden. Moreover, the web-service model will work with any computer equipped with a web-browser.

1.1 Tabulation

Tabulation is the process by which debate teams are paired to debate each other. In a typical tournament, there are between 30 and 150 2-person debate teams, representing between 10 and 40 schools. Over the course of a 5-round tournament, teams must be paired off to debate against each other. Generally, teams will be paired against other teams with the same won-loss record thus far. Teams are also evaluated on “speaker points” and “ranks,” and teams should be paired off so that (for example) the best team with 2 wins and 1 loss debates against the worst team with 2 wins and 1 loss. However, there are a series of restrictions, ranked in order of priority, which must apply to the pairings. Teams should never debate twice, or debate against teams from the same school, etc… For a detailed description of these restrictions, see [2]. These restrictions complicate tabulation so that the process is non-trivial.
1.2 Existing Alternatives

Currently, there are two means of tabulation available to the directors of debate tournaments. The first is doing it by hand—a slow, tedious, error-filled process that delays debate tournaments, requires a great deal of expertise, and invites corruption and favoritism. For schools new to the circuit, hand tabulation is difficult. For competitors, hand tabulation is frustrating because it causes long delays in the tournament and often results in inconsistent application of the stated tabulation policies.

The other alternative is a computer program, the Harvard Tab Program. This program, written as an MS-Access database, is unreliable, often incorrect, and very difficult to use. It requires a specific version of MS Access to work properly, and has not been adequately tested. It is closed source, and frequently produces questionable output. It follows no clear tabulation policy. It is very difficult to use properly, and as a result there are frequent errors committed by the users, causing tournament delays.

These alternatives, for the reasons listed above, do not meet the needs of the APDA community. It is my belief that reliable, well-documented software, delivered as a web service, will better suit the requirements of APDA.
2. System Design

The system is designed as a web service, to be run on a server owned by APDA. The client will use a web browser and log into the APDA server, running the Apache Web Server. The user interface is written in HTML that is dynamically generated using PHP. The web pages contain information about the current state of the tournament, the results of previous rounds, and the pairings for the next round. This information is stored in a MySQL database the PHP code will access. The tabulation algorithms are written in Java; the PHP code will instantiate the appropriate top-level Java Scheduler object, and call its tabulate method as required. The Java code will use its own interface to the MySQL database. Figure 1 below illustrates the overall layout of the system.

Figure 1: System Layout
3. Module Descriptions

3.1 Server

The APDA server will run the standard Linux environment provided by the selected internet hosting provider. The environment must have Apache, MySQL, PHP, and a Java virtual machine installed—but these features are standard for internet hosting providers. Although the system will dynamically generate and deliver many HTML pages, I do not believe the system will require excessively large amounts of bandwidth.

3.2 User Interface/PHP Code

The user interface (implemented as HTML pages) and the PHP code that generate it attempt to create an easy-to-use, responsive interface for the user. Because the user may have only basic computer experience, simplicity and clear design are essential. One of the advantages of implementing this system as a web service is that the interface can be created in HTML, and rapidly modified. Members of the APDA community with some web page experience will be able to contribute to the development of the interface, allowing for the more rapid and successful development of an easy-to-use program. This would not be possible if the system were implemented as a locally run program, because the GUI mechanisms of (for example) Swing are not understood by as many members of the community. This “open-source” development of the interface will allow users of the system to provide feedback about the program, and have that feedback incorporated quickly. One of the flaws with the Harvard Tab Program is that its interface was not created by someone with tournament tabulation experience, and because the source is closed, there are few means by which users can offer feedback on the interface.
3.3 MySQL Database

The MySQL database stores the state of the tournament and the results of previous rounds. The advantage of using a database for this purpose (rather than storing the state and results in memory) is reliability; in the event of a program crash, the system administrator can recover the state and results from the tournament, and either restart the system or at least allow tabulation by hand.

The database will contain four tables per tournament. These tables are described in Table 1 below.

Table 1: Tables in MySQL Database

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>School</td>
<td>A list of school names, unique ids, and payment information.</td>
</tr>
<tr>
<td>Team</td>
<td>A list of team names, school ids, individual debate names, novice status, and seed information for each team</td>
</tr>
<tr>
<td>Team Results</td>
<td>A ranked list of team results. This table is cleared and generated when updated team rankings are required.</td>
</tr>
<tr>
<td>Round</td>
<td>A list of rounds in the tournament. This contains past rounds, with results, and the current pairings.</td>
</tr>
</tbody>
</table>

The School and Team table are created during the tournament registration process. As schools sign up for the tournament, their registered teams will be entered into the Team table, and a new entry created in the School table. New entries in the
Team Results table are created every time the PHP code calls the appropriate Java object. The PHP code will then display those results. New entries in the Round table are created by the Scheduler Java object, when it is called upon to generate new pairings. These entries are then updated by the PHP code, as part of the result entry process.

3.4 Java Code

The java code consists primarily of four objects: the Scheduler, the Team object, the DBLoader, and the ResultsGenerator. The DBLoader loads information from the Round table in the database into a list of Team objects. Each Team object contains the record, speaker scores, and other relevant information for each team. The ResultsGenerator updated the TeamResults table, by running a sort on the Team list. The Team object implements the Comparable interface, so that this sort can be accomplished using a call to Collections.sort. The ResultsGenerator calls the DBLoader to write the team results back into the database, where they can be read by the PHP code. The Scheduler operates in a similar, albeit more complex, fashion. The Scheduler object takes the list of Team objects, and generates a Map object pairing off the teams. The DBLoader then writes this Map object back into the database.

The Scheduler object is actually an abstract class; currently, there are four implementations. The exponential scheduler runs a naïve exponential-time scheduling algorithm. This algorithm is very slow for even moderately size tournaments, so it is used only as a reference. The swap scheduler runs a polynomial time algorithm that creates an initial pairings, and then swaps teams as it attempts to improve the results. The JICOS scheduler runs a polynomial time minimum-weight maximum-cardinality
matching algorithm. The combined scheduler runs both the swap and JICOS schedulers, and returns the better result. The algorithm for the scheduler is described in section 4.

4. Scheduling Algorithm

The scheduling algorithm operates in two phases. First, it creates brackets. A bracket is a group of teams with the same won-loss record. Sometimes, it is necessary to “pull-up” a team from the bracket below (one less win) to ensure an even number of teams in the bracket. If there are an odd number of teams in the lowest bracket (no wins), the worst team in that bracket receives a bye.

The system then pairs each bracket, from highest (most wins) to lowest (least wins). The pairings of each bracket are assigned a weight; the lower the weight, the better the pairing. The weight of each pairings is determined as follows: first, a squared-distance metric is used based on position within the bracket. In a 16 team bracket, the 2\textsuperscript{nd} best team paired with the 15\textsuperscript{th} best team generates a 0 weight. The 2\textsuperscript{nd} best team with the 14\textsuperscript{th} best or 16\textsuperscript{th} best generates a 1 weight. The 2\textsuperscript{nd} best team paired with the 13\textsuperscript{th} best team generates a 4 weight, and so on. Next, a weight is added if one of the four tab priorities is violated. There are maximum of 256 teams in each bracket, so the total cost of all the squared weights will not exceed 256\textsuperscript{3}. We add a power of 256 (3,4,5, or 6, depending on priority) if one of the tab priorities have been violated. This way, a set of pairings where a tab priority has been violated cannot be lower weight than a set of pairings where the priority has not been violated.

The goal of scheduling algorithm is to produce the minimum weight pairings. For each bracket, the weight between each possible pair of teams is computed, and a
weighted, undirected graph created. The minimum-weight maximum-cardinality
matching algorithm from JICOS is then run, and a pairing produced.

In the event that a pull-up is required, a pairings is produced for every possible
pull-up team. An additional weight is added based on that pull-up team’s distance from
the center of its original bracket and whether the team has been pulled up before. The
pull-up team that generates the lowest total weight is chosen, and then the lower bracket
is paired, until all brackets have been paired.

5. Testing and Debugging

The system has been tested in several ways. First, JUnit-based unit tests have
been used to verify that individual Java objects are behaving as specified. Second, a
sample tournament has been run by hand, to qualitatively verify that the pairings are
being correctly generated. Third, both the swap and JICOS schedulers are run, and the
best pairing used. In all cases, this should be the JICOS scheduler, but the swap-based
algorithm provides redundancy in the event of an error. Fourth, an automated tournament
generation system has been created; the system generates results for each round in the
tournament randomly. This allows both manual and automated testing: it speeds up
manual tests, because with the click of a button a tester can generate results for one round
and create pairings for the next round, and then manually verify that the pairings are
correct. The automated testing system runs the result generator, and attempts to detect
scheduling errors and violations of the tab priorities. This is more difficult—it is hard to
verify pairings are correct without attempting to generate correct pairings! Nevertheless,
some automated error detection is possible.
6. Conclusion

The creation of an open-source, easy-to-use, reliable tabulation system should be a boon to the APDA community. Although the algorithms involved are non-trivial, interface design is challenging, and the system has a number of components, this software will be easier to maintain and more reliable than existing alternatives. Using a web-based system for tournament tabulation should help improve the timing and efficiency of APDA tournaments. APDA is fortunate that its rules of tournament scheduling do not require an algorithm of exponential complexity; although the minimum-cost maximum cardinality matching problem is a very complex polynomial time algorithm, it can be feasibly run on the size of tournaments typical of APDA.

7. Acknowledgements

I would like to thank several people for their invaluable assistance on this project. First, Patrick Nichols, formerly of the MIT Debate Team, for the idea of creating a web-service style tab program. Secondly, Jason Wen of the Harvard Debate Team, for assistance testing the scheduling program. Finally, I would like to thank Professor Erik Demaine of MIT for supervising the project.
8. References


