AlgoBowl Software Design

Jack Rosenthal

24 April 2017

1 Introduction

AlgoBowl is an extension to the CS CONNECT web application used in CSCI-406: Algorithms. Students are given a NP-hard problem to solve by the professor and work in small groups (typically of three students) to create and implement a heuristic to get the best solution they can. Once they have done this, each of the groups:

1. Designs an input for the other groups to run and uploads it to the AlgoBowl webpage
2. Downloads all the inputs from the other groups and runs it on their program
3. Uploads their outputs for the corresponding inputs
4. Verifies each of the outputs for their input and marks it as either *Accepted* or *Rejected*

To keep the process flowing, the students are given deadlines they need to complete each of these steps by.

At the end of the assignment, each of the students are given an opportunity to rank their fellow team members' efforts as a percentage. The instructor then downloads these effort percentages and assigns grades to each student.

2 Database Model Design

Figure 1 shows a (Peter Chen style) entity relationship diagram showing the different database models that should be used to implement the software.

Notice the following:

- The attribute `algo_group` as an extension of the CONNECT `User` model specifies the group ID that the student is a participant of. For students not enrolled in CSCI-406, this attribute is simply set to a `NULL` value.

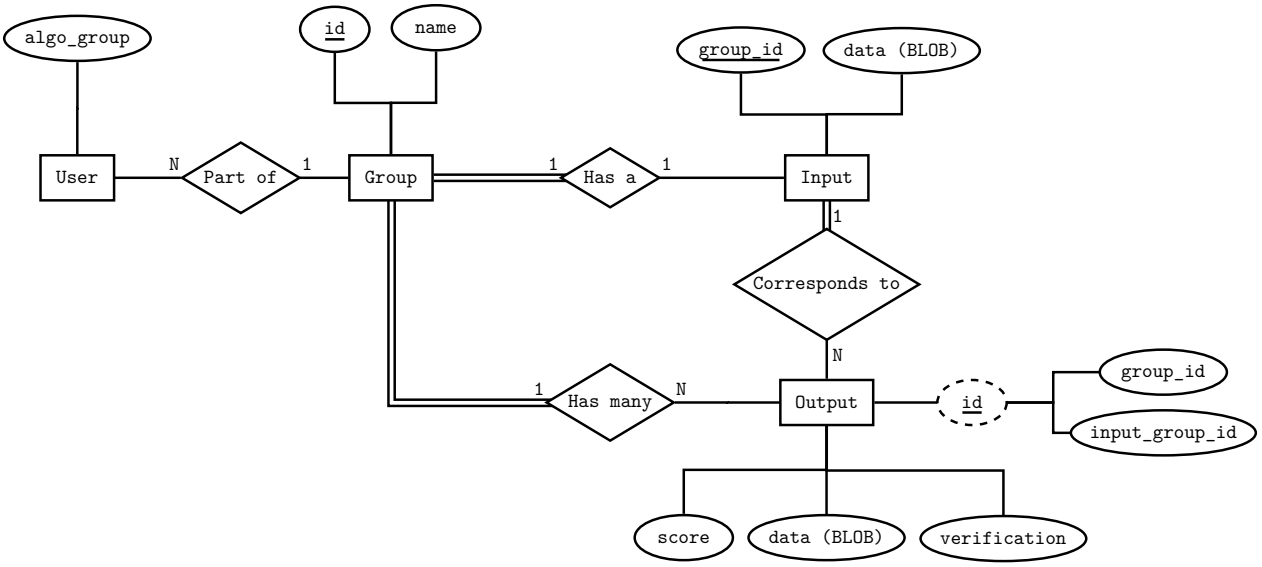

Figure 1: The database model

- The `Group` model relates the numerical group ID to the team's name. Only the instructor may have means of discovering both sides of this relation as the names of the students in each group are not anonymous but their results must be.
- Both the `Input` and `Output` models store the binary data that is uploaded. **This data is not stored on the filesystem!**
- The `Output` model contains both a `score` attribute and a `data` attribute. When an output is uploaded, the first line is separated from the remainder of the file and parsed as a numerical value. This is explained further later on in the controller logic.
- The primary key if the `Output` model is the combination of both the group that uploaded the output and the group that the input corresponds to. There is no `id` attribute.
- The `verification` attribute of the `Output` model is a enumerable type; being one of *Waiting*, *Accepted*, or *Rejected*.

There are two more models which do not have relations to the other AlgoBowl models: the `Evaluation` model (shown in Figure 2) and the `AlgoBowlConfiguration` model (shown in Figure 3).

The configuration table is a Berkeley DB style table containing the configuration options for the current AlgoBowl problem. This table stores weather this problem is a minimization or maximization problem, as well as any other associated data for the problem.

Evalutions

Primary key: (`from_user_id`, `to_user_id`)

Foreign key constraints:

`from_user_id` → `User.id`

`to_user_id` → `User.id`

`from_user_id`	`to_user_id`	`effort`
1	1	35
1	2	35
1	3	30
⋮	⋮	⋮

Figure 2: The Evalutions Table

AlgoBowlConfiguration

`key`	`value`
`problem_type`	`min`
`instructor`	`dmehta@mines.edu`
⋮	⋮

Figure 3: The Configuration Table

3 Controller Design

`GET algobowl`
Authorization: CS CONNECT User
▷ *Generate the rankings page*

When the landing page for AlgoBowl is loaded, a table of rankings is shown to the user. This rankings table is accessible *regardless of weather the current user is a CSCI-406 student* to allow others to look in at the progress of AlgoBowl.

An example table is shown in Figure 4. In this table:

- A ranking of N indicates that the output was not submitted yet
- A ranking of R indicates that the output was rejected
- A ranking of a number with a (W) next to it indicates the output is waiting for acceptance
- A ranking of a number without a (W) next to it indicates the output was accepted
- Groups highlighted in red have at least one rejected output
- Groups highlighted in orange have at least one output they have not submitted yet
- Groups highlighted in green have all their outputs accepted

A few important notes on how this table was generated:

- For a particular input, if any teams tie, they should share the same rank and the ranks of all the teams should offset accounting for the number of tied ranks. For example, if two teams tie for first place, the next place is third place, not second. In other words, if N teams did better on a particular input, then the rank of that team for the input is defined to be $N + 1$.
- When a team has rejected outputs or has not uploaded all of their outputs, they cannot receive a finite value for sum of ranks, because they have not yet completed the competition.
- When a team has not had their output accepted yet, they can be given a rank compared to the others, but the table should indicate it is not accepted yet.
- When two teams tie for their sum of ranks, a reasonable secondary sorting key should be used to show their order in the table. This secondary key (sorted by minimum) is calculated as $R - 2A - W$, where R is the number of rejected outputs for that team, A is the number of accepted outputs for that team, and W is the number of outputs waiting acceptance for that team. In the case that this secondary sort key is equivalent as well, the group's number is used as the third key.

Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Sum of Ranks
Mehtabyte	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	23
Mehta_Palooza	1	2	8	4 (W)	1	2	1	3	3	1	4	2	4	2	2	2	1	3	1	1	3	1	1	53
Dineshire Cats	1	2	8	4	1	6	1	3	3	1	4	2	4	2	2	2	1	3	1	1	3	1	1	57
The Knights Who Say N!	1	8	5	13	1	11	7	10	10	1	12	9	8	7	8	14	1	8	1	1	11	1	1	149
Mehtaphorical Music	1	18	2	1	18	8	13	11	15	20	10	20	3	10	13	8	18	2	1	1	8	13	1	215
Mehta Programming	14	11	6	12	1	13	12	15	12	14	13	11	13	9	10	13	1	18	12	13	14	1	1	239
MehtaBowl	14	7	3	8	1	16	10	17	8	21	11	8	17	14	14	10	1	15	17	17	12	1	1	243
AATMN	14	9	6	11	1	17	7	19	9	14	9	10	10	8	17	12	1	9	17	17	10	16	18	261
Mehta Data	1	12	8	10	15	12	14	12	11	12	16	12	14	11	9	17	15	12	15	1	16	13	20	278
Not Robby	14	15	16	16	1	18	11	20	16	1	14	14	11	16	18	15	1	11	17	17	17	16	1	296
NOT team name	1	21	14	18	1	20	19	21	20	13	20	21	9	19	18	20	1	13	16	1	18	15	1	320
Mehtacritic	14	17	17	17	19	21	17	13	18	19	19	19	16	17	20	7	16	14	17	13	20	1	1	352
Mehtateam	14	22	20	20 (W)	14	22	21	22	21	16	22	22	21	20	21	22	20	21	17	20	19	16	19	452
Mehta-morphosis	1	19	4	3	R	15	9	14	17	11	2	18	1	15	12	11	14	17	1	1	2	1	1	inf
Slightly Less Mad Dog Mehta	R	2	8	4	1	2	1	3	3	1	4	2	4	2	2	2	1	3	1	1	3	1	1	inf
Batman	1	14	R	15	16	7	18	8	14	1	17	15	18	13	16	18	19	16	11	1	13	1	1	inf
Mehtallic	1	10	15	9	R	9	16	2	7	1	2	7	12	6	7	9	13	10	10	1	9	1	1	inf
Never Mehta guy like Mehta	R	13	13	14	1	13	15	15	13	16	15	13	15	12	10	16	1	19	12	13	15	16	1	inf
10 Mehta	20	20	19	19	17	19	20	18	19	18	21	17	20	18	22	21	17	20	14	16	21	R	21	inf
Game of Mehta	1	2	R	4	1	2	1	3	2	1	4	2	R	2	2	2	1	3	1	1	3	1	1	inf
Cohorts In Cahoots	1	2	8	R	1	2	1	3	3	R	4	2	4	R	2	2	R	3	R	R	3	R	R	inf
Mehta Knight	1	16	18	R	R	10	R	9	R	10	18	16	19	R	15	19	R	R	9	R	R	R	17	inf
Mad Dog Mehta	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	inf

Figure 4: An AlgoBowl Rankings Table

- Group numbers should not be shown next to their scores (but instead their group name), as this preserves student anonymity.

`GET algobowl/submission`

Authorization: AlgoBowl Student
▷ *Generate the Team Name Change, Input/Output Upload, and Input Download Page*

`POST algobowl/set_team_name`

Authorization: AlgoBowl Student
▷ *Handle form submission to change team name*

Makes sure the team name is valid and less than 20 characters, and sets the team's name. Redirects back to the inputs page.

`POST algobowl/upload_input`

Authorization: AlgoBowl Student
▷ *Handle form submission to upload the group's input*

1. Converts input to UNIX file format (if in MS-DOS or Mac OS format)
2. If they are replacing a previous input:
   (a) Email all groups who have already submitted outputs and let them know they will need to redo their input and upload again. The text for this email is shown in Figure 5.[1]
   (b) Delete any outputs which have already been submitted
3. Store their input in the database

[1]This is not a catch-all solution for preventing outputs that don't correspond to the correct input (a student could still download an input, the other group changes their input, then the student uploads their output), but it's the best we can do without having students put unique ID's at the top of their inputs and outputs.

```
To: student1@mines.edu, student2@mines.edu, student3@mines.edu
From: algobowl@connect.mines.edu
Reply-To: instructor@mines.edu
Subject: [AlgoBowl] New Input Uploaded for Group {{Group ID}}

Dear {{Team Name}},

Group {{Group ID}} has just uploaded a new input on AlgoBowl and your
team had an output submitted already.

Your old output has been deleted. At your convenience, please download
their new input, rerun your program on it, and upload the new output.

Best regards,

The AlgoBowl System
```

Figure 5: The email to be sent when an output redo is needed

`GET algobowl/input/input_group_{group_id}.txt`

Authorization: AlgoBowl Student or AlgoBowl Admin
▷ *Download input for group* `group_id`

If Group `group_id` has not uploaded an input yet, this page should abort with a 404 error.

`GET algobowl/all_inputs.zip`

Authorization: AlgoBowl Student or AlgoBowl Admin
▷ *Generate a ZIP archive of all the inputs for the user*

`POST algobowl/upload_output`

Authorization: AlgoBowl Student
▷ *Handle form submission to upload the group's output*

Takes a parameter of the input Group ID the output corresponds to. This should:

1. Converts the output to UNIX file format (if in MS-DOS or Mac OS format)
2. Strip the first line off the output and convert it to a numerical type. This is the `score` attribute. If this process fails, it should raise an error to the user.
3. Store the remainder of the output as the `data` attribute
4. Set the `verification` attribute to *Waiting*
5. Store their output in the database

`GET algobowl/verification`
Authorization: AlgoBowl Student
▷ *Generate a page with output download links and a button to accept or reject each of them*

`GET algobowl/output/output_from_{from_id}_to_{to_id}.txt`
Authorization: AlgoBowl Student with Group ID `to_id` or AlgoBowl Admin
▷ *Download output from group `from_id` to group `to_id`*

`GET algobowl/all_outputs_to_{to_id}.zip`
Authorization: AlgoBowl Student with Group ID `to_id` or AlgoBowl Admin
▷ *Generate a ZIP archive of all the outputs to group `to_id`*

`GET algobowl/evaluations`
Authorization: AlgoBowl Student
▷ *Generate student evaluations page*

`POST algobowl/evaluations`
Authorization: AlgoBowl Student
▷ *Set AlgoBowl Evaluations for a certain student*

This controller method should verify the submitted evaluations sum to exactly 100, and raise an error otherwise.

`GET algobowl/setup`
Authorization: AlgoBowl Admin
▷ *Geterate AlgoBowl Setup Page*

`POST algobowl/setup`
Authorization: AlgoBowl Admin
▷ *Process a new AlgoBowl setup*

This method should:

1. Reset AlgoBowl, clearing any previous data
2. Set up the new AlgoBowl with the specified teams and configuration

`GET algobowl/verif_editor`
Authorization: AlgoBowl Admin
▷ *Generate verification editing page for the admin*

`POST algobowl/verif_editor`
Authorization: AlgoBowl Admin
▷ *Process a verification edit for the admin*

`GET algobowl/eval_admin`
Authorization: AlgoBowl Admin
▷ *View evaluations*

`GET algobowl/evaluations.csv`
Authorization: AlgoBowl Admin
▷ *Download a CSV file with the evaluations*

4 Interface & View Design

4.1 Tabs

For students, the tabs should read:

`Rankings | Submission | Verification | Evaluations`

For admins, the tabs should read:

`Rankings | Submission | Verification Editor | View Evaluations | Setup`

For users who are not logged in, they should be able to view the rankings page, but instead of the tabs there is a message:

You are not currently enrolled on an AlgoBowl team. If you think you have received this message in error, please contact your instructor.

4.2 Rankings

The rankings page should view like Jack's prototype:
`https://mastergo.mines.edu/cgi-bin/algotbl.cgi`

Note that the view for this page from the old AlgoBowl will likely not be anyhow salvageable, as it is far different from Jack's prototype.

4.3 Submission

The submission page should be like the one on the current AlgoBowl:
`https://mastergo.mines.edu/csconnect-oldprod/algobowl/submission`

It should also include a link to download all inputs as a ZIP file.

For the admin user, do not show the Input Upload or Change Team Name sections.

4.4 Verification

The verification screen should look like the one on the current AlgoBowl:
`https://mastergo.mines.edu/csconnect-oldprod/algobowl/verification`

It should also include a link to download all outputs for that team as a ZIP file.

4.5 Evaluation

The evaluation screen should look like the one on the current AlgoBowl:
`https://mastergo.mines.edu/csconnect-oldprod/algobowl/evaluations`

4.6 Verification Editor

The verification editor screen should present the admin with a matrix of verifications (text boxes containing A, R, or W) and allow the admin to edit the verifications and resubmit the page.

4.7 View Evaluations

The evaluations page should show a table of all evaluations to the admin as well as a button to download a CSV file.

4.8 Setup

A warning should appear on this page:

Submitting this form will setup AlgoBowl for a new semester, deleting all current AlgoBowl data

The admin should be able to design teams and select between a minimization or maximization problem.