The Rutgers TA Assignment System (RTAS)

Wael Ayadi, Nick Romano
CS526 - Spring 2018
Rutgers University
Piscataway, NJ, USA
Email: wael.ayadi@rutgers.edu, nick.p.romano@gmail.com

***Abstract*— The Rutgers TA Allocation System (RTAS) is a web application that takes a list of Teaching Assistant (TA) applications as input, processes them using a 3D-Matching algorithm, and allows the user to provide feedback to the algorithm through human-computer interaction in order to extract a collection of approved TA applications that the user is satisfied with. Each TA application is a 3-tuple from the cross-product of three entity sets: the TAs submitting applications, professors, and courses. A generality of this use case is the well known 3D-Matching problem, which is an NP-hard problem where the goal is to find an optimal matching between TAs, professors, and courses. The RTAS will use an approximation algorithm to perform its processing, and it will provide a visual interface that will allow the user to provide feedback to the approximation algorithm in hopes of returning a better output.**

I. Project Description

The a great number of Rutgers departments allow for graduate students to apply to become Teaching Assistants (TA), where they will be paired up with a professor from that respective department who will act as their mentor, and they will be assigned a course from that respective department that they will be helping to teach. Each TA can only be assigned to one professor and one course, and every professor wants to have a TA that will assist him in his research. Therefore, we would like the maximum number of TA applications approved. However, this is not easy to guarantee, since there can be conflicting TA applications (e.g. two TAs can apply to work under the same professor). As a result, this TA assignment process often tends to be very difficult and tedious. This is because this problem is an example of the 3D-Matching Problem, which one of the well-known NP-hard problems in computational complexity theory.

This is a 3D-Matching problem, where the three entity sets that are being matched are TAs, professors, and courses. The goal is to find the largest **matching**, where a matching is a collection of TA applications that do not conflict with each other. Because this problem is NP-hard, the fastest known algorithm for finding a maximum-sized matching has an exponential run-time, which means that it is too slow to be feasible to use in practice. There are approximation algorithms that we can use to derive solutions for the 3D-Matching Problem that are at least a fraction as good as the optimal solution.

However, the approximation algorithm can output solutions that are quite varied, and normally do not have the means to accommodate for a user's preferences in the matchings. For example, a professor may have had a good experience with a certain TA in the past and prefers that specific TA to work with him over other TAs. However, forcing his preference onto the final matching may result in a final matching that has a smaller size than a final matching without his preference enforced, which can have negative consequences on other professors or courses.

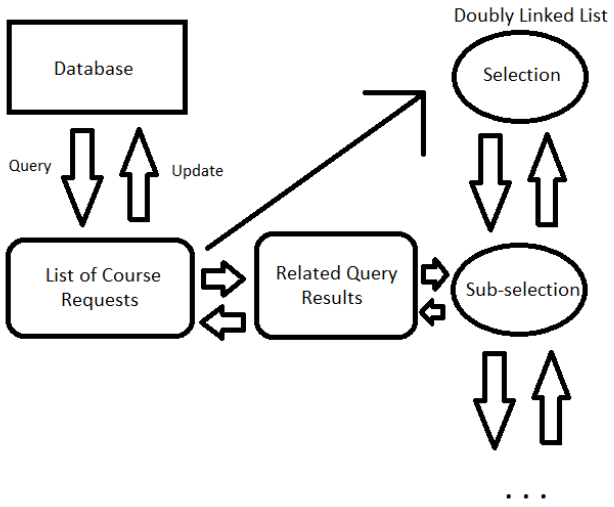

Fig. 1. Internal Data Structures

We propose the Rutgers TA Assignment System (RTAS), which a system that will provide the means to incorporate human-computer interaction into this TA application approval process. The system will provide a visual interface, which will allow for user preferences to be entered. Afterwards, the system will display how the entered user preferences change the quality of the final matching.

A. Stage1 - The Requirement Gathering Stage.

B. Stage2 - The Design Stage.

This project will use multiple technologies for both the front-end and the back-end. For the back end, we plan to use an SQL database to store the Course, Professor, and TA data. Java servlets will be used for for code execution. For the front-end, we plan on using Angular and Javascript for the layout and D3.js for the visual representation of the 3D matching. We may use jQuery for SQL query manipulation that will serve input to the 3D matching approximation algorithm.

As shown in Figure 1, the back-end operations will be driven by queries to the MySQL database as well as the 3D matching approximation algorithm. The results of the matching algorithm will be fed into the D3.js API that will generate a visual representation of the matching. The user will be able to visualize and manipulate the 3D matching to generate different instances (i.e. solutions) of the matching. When the user selects certain nodes on the interface, such as a specific student or class, those nodes will reference a linked list of triples for which the node belongs allowing the user to quickly visualize local interaction between nodes. The current state of the instance of the 3D matching representation will be stored in a doubly linked list that will allow for progression and regression of the visual interface. Using a linked list will allow the user to navigate through different instances (solutions) quickly without having to re-run the approximation algorithm.

C. Stage3 - The Implementation Stage.

D. Stage4 - User Interface.

There will be a single primary view for the user interface. The main concept of the user interface is to allow the user to interact with and affect the instance of the TA/Professor/Course matching. This view will contain a representation of the current TA/Professor/Course matching and will allow the user to view the triples in the current instance of the maximal matching. The exact visual representation of the matching will be determined as the project progresses. Several options for interactive visualization are being considered. A graph representation of the matching (where the nodes represent TA's, courses, or professors) may be the most efficient method of providing a visual representation of the entity relationships. The group may consider adapting a hierarchical graph or possibly a radial graph representing all possible sets of TA/Professor/Course triples where the edges between nodes represent possible triples in the maximum matching. The user will be able to interact with the graph by clicking (or hovering) over a node to see the result of the matching. The user will also be able to interact with the current instance of the matching by assigning (or un-assigning) triples. User feedback will trigger the algorithm to recalculate the maximal matching under the user defined assignments and the corresponding visualization will be displayed.