Yo I’ll Solve It: Brainstorming

# 1:1 Scheduling Problem (Max Bipartite Matching)

## Algorithms:

### Ford-Fulkerson:

This is bad because it only allows for a simple version of the Maximum Bipartite Matching Problem, when we really need a more flexible algorithm that allows for solving variations of the problem. (First question in the questions section)

<https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/matching.pdf>

<https://www.w3schools.com/dsa/dsa_algo_graphs_fordfulkerson.php>

### Simulated Annealing:

<https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html>

<https://www.cec.uchile.cl/cinetica/pcordero/MC_libros/NumericalRecipesinC.pdf> (Cp. 10)

### Hill-climbing:

<https://en.wikipedia.org/wiki/Hill_climbing>

## Questions:

How does the problem get modified by introducing other constraints, for example, when talking about TA scheduling how does the convenience of a time get factored into the problem?

A whiteboard with a clock and arrows

Description automatically generatedA whiteboard with writing on it

Description automatically generated

Additionally, what else do we need to know from the user? From the user we need rankings of convenience per student per timeslot. We need to know how important convenience is (should we schedule less time slots to maximize convenience), this could be a relative importance between all the constraints. How do we score convenience, would we rather have 9/10 students at max convenience and 1/10 at min convenience or, 8/10 students at max convenience and 2/10 at middle convenience?

We want to use a method of weighing the scores of an algorithm such as hill climbing will extra parameters set by the ProblemHaver.

What about this version tree/edit history tree, how much information is too much information? When should an edit be added to the tree? How much does the solution need to change for an entry to be made in the tree? Should the user choose when to make an entry into the tree, like how git works?

We don’t need to show the user everything we store, we can group edits so that they appear as a single node on the tree. When the user clicks to a node, somehow present the individual edits so the user can get to the specific spot they want to go.

About the edit section, should it be possible to regenerate whole solutions, what about partial solutions? Yes. Should you be able to modify the original weights for the parameters? Should the software suggest changes to the weights based on how the ProblemHaver modifies the solution? Are there other ways to represent the solution so that there is a more natural way of modifying the solution? For example what if I want to meet with student A before student B for whatever reason how would that information be passed to the software? Should you be able to add more parameters? Should the software try to suggest extra parameters that might help?

These ideas are interesting and we will attempt to experiment with multiple of them, also many of them I would have no clue even where to begin implementing.

The software should produce a bunch of fake data and solutions that is easy to understand for the user and also can help the program and the user agree on the problem that is being asked?

Yes certainly, how could someone ever fully grasp what they are asking a computer to do by only inputting a few numbers or formulas without knowledge of the inner algorithms. We want to implement the strategy design pattern allowing the software to choose what strategy it deems best fit for the problem without the user seeing any of that. A few small examples would help the computer understand what the user needs from a solution of the problem.

During the initial solution selection process should we do a sort of eye test type thing, where the ProblemHaver picks their favorite solution of a few and then the program gives the option to select a few different solutions based on that pick? If you did that, when do you stop, how many times do you repeat the selection?

This is closely related to previous set of questions, this seems like a good idea and how many times you repeat could be left to the user or discovered via experimentation.

## Ideas:

There should be a way to reuse questionnaires with the same weights and with a new list of consumers. There should be a way to create templates for questionnaires. These templates should have a name and description and should be shareable to a database so more people can use the same template.

# Feb 20

1. Status report:
   1. Tasks completed:
      1. generating fake data
      2. reading the fake
2. Next steps
   1. Implement for Fulkerson
   2. Sketch more interface ideas
   3. Personal Professional Profile to publish/document your work <https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-profile/customizing-your-profile/managing-your-profile-readme>
3. Can Stewart do TA scheduling in the Fall with this stuff?
   1. Can get TA availability from questionpro
      1. Need to be sure this project has ingest that works for csv in that format
   2. Need to be able to input the empty schedule slots
   3. Ingests a configuration
      1. To inform scoring
   4. Need to be able to output a possible solution
   5. To generalize (for these 2 problems)
      1. A problem has
         1. Producers
            1. 1:1 – Stewart’s available slots of time
            2. Ta scheduling – the weekly shifts
         2. Consumers
            1. The students who wish to meet with stewart
            2. The TAs who wish to work shifts
         3. Configuration/Parameters
            1. Penalties or rewards
4. How do we give a user an interface to the penalizers/rewarders of hill climbing or similar approaches bc what is like “1000” vs”10”?
   1. Highly possible that parameters that don’t best represent their intent anyway produce a favorable solution
   2. What if we require introduce 1 param at a time
      1. Ta scheduling
         1. First step is only assign possible shifts
         2. 1st refinement: try to spread the 261 experience
         3. Next refinement: reward 2 consecutive shifts for a person (penalize 3)
      2. This requires ordering them correctly (bc we don’t permit subsequent refinements to break previous)
         1. But we could let them explicitly say I don’t care as much about the previous step
      3. One way to make the user do this is by building constraints like nodes in a tree, and the user can have different branches that try different things