Bryan Tsai

Using Network Flow Reductions to Solve Scheduling

Take the case of ScheduleDem (misspelled as ShceduleDem in the repo).
In this instance, the primary purpose of the project is to create a scheduling app on android to solve scheduling instances for organisational tabling, and possibly other applications.

In the case of "tabling," there are some number of table sessions, and some number of volunteers or organisation members, who are able to perform tabling for some list of table sessions. Furthermore, there would also some minimum number of volunteers needed to table each session. This means that you can't have a tabling session be empty.

The intuition, in this sense is that organisation members or volunteers can be viewed as "flowing" into tabling session slots.

To that end, one of the simplest models for solving this scheduling problem in the sense of each filling out each table session would be to have a node for each member/volunteer, with a pipe leading from the source to said node for that member with a capacity equal to the number of tabling sessions that they are able to work at. Next there would be a node for each table session, and for each member/volunteer, a pipe of capacity 1 going from the node for that volunteer to the node of a session that they are able to work at, for all sessions that they are able to work at. Finally, for each node representing a tabling session, there would be a pipe leading from that node to the sink, of capacity equal to the tabling capacity (in terms of working members) for that session.

This model would be described by the included image. [Figure 1]

Finally, to solve this problem, simply run a network flow solver and check if the maximal flow through the network is equal to the sum of the capacities of the pipes leading from each tabling session to the sink. If it is the case that the maximal flow is equal to the sum of the capacities of the pipes leading from each tabling session to the sink, then all tabling sessions would have the minimal number of volunteers needed to table each session since such a maximal flow would mean that each tabling session node is contributing it's maximal flow value, and each tabling session node only has incoming flow from the nodes of volunteers who are able to work at that session.

As for finding the scheduling solution, simply look at the flow configuration given by the flow solver for the maximal flow and assign volunteers to sessions according to which volunteer → session pipes were active (had flow running through them).

Further notes.

User interface stories:

-I want to get as much labour as possible for my tabling
Solution:
After running the solver to find the minimum requirements, simply assign additional volunteers to tabling sessions based on availability until remaining slots are filled.

-I want each member to work 5 tabling sessions.
Solution:
Modify the network creation algorithm such that the pipe leading from the source to each volunteer node has capacity equal to 5 or some number X that represents how much each volunteer has to work.
-But what if I want each member to work at least 5 tabling sessions, but not necessarily stop at 5
After running the 5 tabling session version of the network creation algorithm, and solving that instance, feel free to assign additional volunteers to sessions.

-I want to take into account that some sessions may have special jobs that can only be fulfilled by some members but not others
Solution:
Modify the network creation algorithm such that there are two "buffers" between the volunteer nodes and the tabling session nodes.
The first buffer shall simply be a node that counts how many jobs each volunteer is able to fulfil for a specific tabling session (Default 1). The second buffer shall be labour pools for each special job for a specific tabling session, where the connection between the first buffer and the second shall be according to which volunteer is able to do which job for a specific tabling session (regular or special). [Figure 2]