Student-Job Placement Algorithm

Harish Karnick with Devanshu Somani

January 8, 2018

What we aim to implement

- Our basic aim to implement a new system is the inclusion of students' priority lists for jobs into the scenario and not only the jobs' priority lists of students.
- The relative preference given to both of them can be controlled using the SPO's policy.
- Also, many problems arise due to the presence of a large number of slots during the placement process. Everyone rushes for Day1-Slot1 but as all companies can't come in such a small time period, many students have to compromise on their preferences, just to be on the safe side. Thus, we also aim to decrease the number of slots by enlarging each slot's time period.

A brief idea of the proposed system

The Algorithm of student-job placement that we came up works as follows :

Let the Companies have with them a total of M jobs. Let J be the set of jobs.
$J = \{j_1, j_2, ...j_M\}$
This straightaway requires all the companies to provide the number of students that they would be hiring well before the end of the slot. Also, each of the jobs will have a list of students arranged in a definite priority order associated with them. It would mean that the job can take any of the students from that list, but its priority would be given to the top student in their list. And since many companies would want the same students, the list of each of these jobs should be large enough to avoid vacancies.

A brief idea of the proposed system - Continued

Similarly, all the students applying for the jobs should be assigned a number (for ex- their Roll No.) and let the total students be N. Let S be the set of students.
$S = \{s_1, s_2, ...s_N\}$
Just like the jobs, the students would also be required to submit their preferences of the respectively applied jobs in order well before the end of the slot.

Now, the algorithm we developed creates a matching of the set S with the set J. Let the student associated with j_i be at r_i rank in the job's priority list and the rank of the job j_i in that student's priority list be t_i.
What our algorithm does is that, it minimizes the value of

$$K = w_1 \sum_{i=1}^{M} r_i + w_2 \sum_{i=1}^{N} t_i$$

These w_1 and w_2 are constants and represent the relative preference given to the students' and the companies' choices.
Both of them can be set according the the SPO and can be set as its policy.

Advantages of the proposed system over the current system

- One of the most obvious advantage is that this system is flexible and matches jobs with students all at once (for a slot). Therefore, if a student gets more top spot in two or more, the student below him/her in the vacated job's priority list would also be able to check his/her priorities, even if he/she is already associated with some other job.
- Also, a salient feature of the proposed system is the flexibility of relative preference given to jobs' and students' priorities. A high w_1 would mean greater preference to the former and low w_1 (or high w_2) would mean greater preference to the latter.
- Also, once w_1 and w_2 are specified and all the priority lists are fed to the algorithm, the algorithm would come up with the possible allocation by minimizing K as shown in the previous slide.

Let us take some examples to understand how will this work.

A few examples - Problem with Large Number of Slots

Let us first understand the problem created by the presence of a large number of slots.
Consider the following scenario :

Let s_1 and s_2 be two very bright, almost equally able students who are brothers and both determined to work in a particular Company C together. But C couldn't come in first day due to some issues and therefore, to stay on the safe side, s_1 also filled in for some backup job being offered on Day1-Slot1. But, s_2 being determined to work in C did not do that.

Since s_1 was a bright student, he easily got selected for the backup job on Day1-Slot1. s_2 waited for C to come up, and was selected for a job offered by C. But, s_1 already had got a job and thus couldn't do anything as he is out of the placement process.

This example shows how even after being equally abled to qualify for Company C's job, s_1 doesn't get it. Not only they are separated, will have to work apart, but also s_1 didn't get what he deserved/wanted.

The situation could have been much better if Company came in the same slot as the other backup job. Thus, the number of slots should be decreased by increasing each slot time to like 3-4 days.

A few examples - What the Student Wants

Now, let us look in this totally job-favouring scenario which is somewhat like the current system but has significant merits over it :

Let $w_1 = 1$ and $w_2 = 0$.
Consider the following job and student preferences in decreasing priority order :-

j_1 - s_1 s_2 s_3
j_2 - s_2 s_3 s_1
j_3 - s_1 s_2 s_3

s_1 - j_1 j_2 j_3
s_2 - j_3 j_1 j_2
s_3 - j_1 j_2 j_3

The allocated job with students in the jobs' priority list would look like this, where the dashed text represents job allocation that should be ideal and the red text represents what the current system followed at the SPO would have done :

j_1	- s_1	s_2	s_3
j_2	s_2	- s_3	s_1
j_3	s_1	- s_2	s_3

In the current system :
As s_1 has two top spots, he/she can choose from j_1 and j_3. He will choose j_1, thus creating a vacancy in j_3. Now, since s_2 is already associated to j_2, the current system would overlook s_2 in j_3's priority list to ultimately give it to s_3, not knowing that s_2 wanted j_3 over j_2 and j_3 wanted s_2 more than s_3. Also, s_3 wanted j_2 more than j_3.

How to implement

- A provision should be made for all the students that apply for the jobs, where they could all submit their preferences of companies in order (out of the ones which they have applied for)
- The time duration of each slot should be increased to 3-4 days so that students and jobs both get better results.
- The companies should before hand provide the number of students they would be hiring in each slot.
- Any company hiring more than 1 student would create jobs with identical student priority lists(say n). For any student who has applied for this company with some rank(say r) in his/her priority list, all those n jobs provided by the company would carry the same rank r in the student's priority list. A notable fact here is that, the next rank in the student's priority list would be $r + 1$ and not $r + n$ as all the jobs are just replicas of each other and represent a single company.

Some Extra Information

- The Algorithm would work only if w_1 and w_2 are both positive. Setting any one of them to be zero may give multiple possible allocations and would different algorithms to work.
- Each replica job(for a company hiring more than 1 student) should have the same student priority list. Also, the students would give a rank list for companies and not jobs. Hence, the student priority lists should be converted efficiently as mentioned in the last slide.
- Practical problems due to a longer slot duration, like stay for company interviewers and executives and over-anxiety in some students would be faced.

Thank You