Scheduling Fastenal Company Employee Training

Team A5

October 2015

1 Definitions

1.1 Schedule

For out purposes, we will define a schedule S as a 5-tuple of the form $S = (P, C, T, R, M)$, where P is the set of all instructors, C is the set of all classes, T is the set of all class times, and R is the set of all available rooms. Finally, M is a set of 4-tuples of the form (p, c, t, r), which is the assignment of instructor p to teach class c at time t in room r. Components of S will be referred to as $P(S), C(S), T(S), R(S)$, and $M(S)$.

1.2 Complete Schedule

A schedule S is called complete if, for every course, there is the exact number of required sections of that course.

1.3 Viable Schedule

A schedule S is viable if, for any subset A of M, the elements of A are not in conflict of each other.

2 Method

Our attempted method consisted of defining the search space of all possible schedules as a tree, and then repeatedly pruning that tree until it was feasible to conduct a depth-first search of the tree for all viable schedules. Once all viable schedules were found, they would each be evaluated by a fitness function, which assigns each schedule a numerical value between 0 and 100 corresponding to how well the schedule satisfied the desires of each person.

Our initial method of pruning was to create a set of viable classes, as triples (p, c, t) in which instructor p could teach class c at time t. This reduced the number of children at each node by many orders of magnitude.

3 Evaluation

We built a function $g(S)$, to determine the effectiveness of a schedule. We used the four guidelines given for; A, the instructors satisfaction of the classes they were teaching, (based on what class it was, and the time it would be taught), B, The workload and prep minimization of the professors, C, The room efficiency, and D, whether the teachers would have to switch rooms when teaching 2 classes in a row.

$$g(S) = w_A A + w_B B + w_C C + w_D D$$

The weights could be chosen based on what importance you give each factor, however we created some weights based on what we thought it should be. We thoughts that A and B were around the same amount of importance, with C being more important than A or B, and D being the least important. We ended up with what is shown in equation 2 below.

$$g(s) = 0.25A + 0.25B + 0.4C + 0.1D$$

A we defined as the satisfaction of the teachers. For each section taught, the total preference would increase by 1 point if the professor prefers to teach that course, as well as 1 point if they like the time of the class. This gives each section a possibility of having 2 points, if the professor likes both the time and course. With 116 sections, this gives a total possibility of 232 satisfaction points. This gives us an equation to find our satisfaction factor using equation 3 below.

$$A = \frac{\text{total preference points}}{232}$$

B was defined as the work load (prep time). The more courses a professor teaches, the more work they have to prepare for the courses. A senior can teach 3 sections while a junior can teach 4 sections. The most efficient would be if they only taught sections of one course, while the least efficient would be if they taught all different courses. We distributed 0-6 work points to each teacher based on how efficient their workload would be, as shown in the table below.

Number of Courses	Junior	Senior	Manager
1	6	6	6
2	4	3	NA
3	2	0	NA
4	0	NA	NA

Also the manager is guaranteed a 6 (as he cant teach more than one class)****** This gives a maximum workload score of a 6 to every teacher. With 10 Juniors, 28 Seniors and 1 manager, for a total of 39 teachers, this gives a total workload score of 234. This workload score of B is found by the equation below.

$$B = \frac{\text{total work points}}{234}$$

C was defined as the Room efficiency. This could be open to interpretation in different ways, however we chose to say if you used less rooms overall in the schedule, it would be more efficient. The rooms not included could be used for other things (offices, study/tutor rooms, research, administrative purposes, campus resources, or rented out). With 116 classes, and 8 hours in the day, the most efficient would 116/8, this equals 14.5, so if only 15 rooms are used that would be maximum efficiency, and would get a score of 100%. 25 would be the least efficient (using all the rooms) and would get 0%, so we made an equation to make a linear relationship between these two points, shown in the below equation.

$$C = 1 - 0.1(\text{ rooms used } - 15)$$

D was defined as the Transition Efficiency. This is saying that if a professor teaches 2 classes in a row, it is more efficient if they teach in the same classroom, than switching classrooms. So each time a professor has to teach 2 classes in a row, add 1 point to the transition score, and 1 point to the efficiency score if they stay in the same classroom. Using how many transition efficiency points we can get D shown below in equation 6.

$$D = \frac{\text{efficiency score}}{\text{transition score}}$$

4 Conclusion

While our methods of pruning the search tree were not conclusive in reducing the search to a manageable level, we did implement a functioning prototype in C. We chose C as it would be the most time efficient implementation. Our implementation could successfully create working schedules on smaller data sets. However, when run on the total data set, its asymptotic run time was still too large to be practical.

However, late in our implementation, we came up with several ways in which our search could be rapidly sped up. The main way in which we decided to do this was to sort the viable classes by a heuristic to estimate how well they would perform in a finished schedule. This would allow us to find a decent schedule more quickly, then iterating through, and sorting many mediocre schedules.

We also came up with several ways to fast-forward through our search by caching results which viable classes are incompatible with each other, allowing us to skip nodes until we found one that is either compatible, or non-calculated. By some rudimentary calculations, that would provide massive speedups over our naive solution.