CS 2951-O
zhoffman
zling1

Final Report: Project II Employee Scheduling

Introduction

Our original formulation of this constraint problem for part I was almost identical to the one presented in class. Thus, we didn't need to make many changes in part II to our constraint model. Most of our work in part II focused on our DFS search strategy. Our final strategy of random restarts and random explicit evaluations lead to search that was far faster than the builtin search strategy used by part I.

Random Restart Search Phase Design

We initialize a random restart strategy with a fail limit of 1000 and a growth of 1.01. We initialize our seed to 1 and increment it by 1 with each restart. With each restart, we randomly initialize valuations of the hours and shift domains. At each step, we use minimum domain size variable selection with ties broken randomly. We then select the value with the highest valuations (which was initialized randomly). Before choosing the approach of random initialization, we tested hardcoding the valuation of different variables. We noticed that their valuations made a drastic difference in program speed. Since the domain size was so small, we figured that we'd find a good initialization quickly with fast restarts. We chose a very small growth size to allow quick iterations through seeds and fast restarts.

Other Attempts

We also tried using local impact, success rate, and smallest value to decide between values, but these timed out for over half the cases.

Assessment of Schedule Quality

The "quality" of a schedule is largely determined by business needs. Certain jobs like security need people on the job at all times. For this type of job, schedule quality is highly dependent on time coverage. Other businesses have large fluctuations in required workers day to day. For example, sports stadiums require far more workers on the days of a big game. For this type of job, the schedule quality will be dependent on how well the schedule maximizes workers on prioritized days. Some companies give their employees holidays and weekends off and a good scheduler will be able to account for these requirements. In the next section, we discuss how our scheduler can handle these scenarios.

Business Analysis

Our model doesn't explicitly set start and end times, so those can easily be assigned to appease any arbitrary business strategy. We can also change the minimum required workers per shift without much issue. Take the case where we always want at least one person on the job. To this, we need at least 8 hours covered for each shift. One solution is to require at least two people working each shift. As the minimum hours worked per person is 4, two people cover at least 8 hours. Another approach we can is to adjust the minimum demand per shift chart to 8 hours per

shift. If we wish to extend this further and have *N* number of workers always on the job, we can require 2*N* people per shift minus the number of people working 8 hours.

If we wish to increase the quality of life of our workers and regularly require days off. We can add constraints to our model that require a certain number of days off each week. To do this we can count the number of off shifts a worker has each week and require it be greater than some amount. This also produces the side effect of increasing the number of people working on other days to make up for the time off as shown in the table below.

Min # of Off Shifts	3	5
Average # of 0ff Shifts (measured per day per employee)	0.37244895	0.36224493
Average # of Working Hours (measured per day per employee)	4.102041	5.102041

If we want to adjust production on the business sides by having our employees working longer or shorter shifts, we can change the domain of the hours worked variable. This approach also leads to employees having more days off. If we need more people working on certain days, we can change the minimum number of required shifts for those days. If we wish to enforce holidays, we can add a constraint that requires the total number of shifts on holidays to be 0. We can either do this by counting or summing shifts, but summing shifts is more efficient.

If an employee needs to take a day off, we can easily check the schedule to see which employees are available to fill their shift. If we wish to always have a substitute employee to fill a shift, we can add a constraint that counts the number of off employees who have 4-8 hours to spare and require the total be greater than 1.

Conclusion

Overall, our scheduler is pretty flexible to business needs. Due to the flexibility, our scheduler can produce high quality schedules for a variety of businesses.

Names, Logins, and Leaderboard Names:

- Leaderboard Name: KingKrab
- Name: Zachary Hoffman, Login: zhoffman
- Name: Zihan Ling, Login: zling1