Project 2 Writeup

Names: Natalie Lindsay, Purvi Goel
Screen name: pizza_ball

When approaching Part 2 of the project, we found that several changes were required in order to optimize the model. Our main goal was to reduce the “parameter space” of variables that the solver would have to test. With a smaller landscape to explore, the solver would terminate faster.

Phases

First, we found that it was helpful to split the constraints up into phases. We used two strategies: first, we created a separate phase for every week. The idea here was that the solver would only have to solve for a 7 day period at once, especially because most of the problem’s constraints (daily work, number of night shifts, etc) would reset at the end of the week. Second, we split up the first week into two phases: a *training* phase for the first 4 days, and a *normal* phase for the remaining 3 days. This is because the output of the first 4 days could help constrain the solution for the rest of the week, and was a sufficiently small enough time period that it should be solved first.

Constraints

In addition to the basic constraints posed by the problem statement, such as minimum daily hours and maximum night shifts, we also chose to limit the times within a single day that a worker could begin their shift. This would reduce the number of variables that the solver had to test, because most times within the day were off limits. Rather than having a worker start their work at any time in the day (00hr - 24hr), we imposed constraints that demanded a worker start at the start of the shift (00:00, 08:00, 16:00).

Solver

We found that it was helpful to use a VarSelector to help with tie-breaking. We gave it three different options: select smallest domain size (limit the number of options at the start to reduce the parameter space), select largest impact (we want variable choices to have a large effect on the parameter space so the solver knows if a solution is right or wrong faster), and choose randomly. We used trial and error to try different options and orders before we settled on this one, as it produced the fastest solutions.

We also used a ValueSelector to order the variable options for the duration of the shift. We thought it would be better to more often choose a length of shift that was in the middle of what was allowed (6 hours), because we didn’t want to bias the selector to always choosing the longest or shortest shift possible. The ordering we gave is somewhat arbitrary, but the important thing is not having 8 or 4 first.

Analysis: Business

As a business owner, one of the goals one would have for scheduling is to maximize production while still adhering to the constraints on how much a worker can work in a certain time period. This probably translates to maximizing each worker's hours, so that there is never the possibility of a staff shortage on any shift. However, another consideration for the business could be to have workers that are not exhausted / overworked, so you wouldn't want to go too far in maximizing the number of hours worked. I think the existing constraints, like the number of night shifts in a row constraint, would most likely prevent that situation, but it's still something to consider. One constraint that could be introduced to achieve this goal of maximizing the production would be to always have workers work an 8 hour shift - right now it can be anywhere from 4-8 hours, and there is no constraint that says that there always has to be someone on the shift at any given time. This could be problematic if Employee 1 and Employee 2 both started at 8am and worked till noon, but then after that there was no one on the shift. The other way to fix this would be to constrain that there is always at least a set number of people on a shift at any given time. To assess the quality of the schedule from a business perspective, you probably want to first check that there are no staff shortages on any of the shifts. You definitely want there to always be at least one person working and you probably know that certain shifts are busier than others, so those should ideally have more workers. At the same time, a business probably wants to minimize the amount it has to pay workers while maximizing production, so there's most likely an ideal number of workers for any shift that is enough to keep production up while not over scheduling that shift.

Analysis: Workers

Workers in the company will have different goals than the business owners. Rather than maximizing the amount of production, they will likely value the quality of life and amount of work more. Workers will want to ensure that they are not overworked and the shifts match their preferences. We would suggest adding new constraints based on the amount-per-week that every individual would ideally like to earn, at a minimum. Workers would want to maximize the amount that they earn, but keep their shifts fitting under the maximum hours required per week. From the workers' perspective, a "quality" schedule would be one where the shifts are staggered in a way that would not overwork them. For instance, though *minConsecutiveWork* might be 4 hours, a worker might find it more ideal if the solver tended towards solutions that gave them consecutive hours closer to 4 rather than higher. Additionally, if a shift is known to be busier, an employee wouldn't want to be the single person on that shift. A higher-quality schedule would have more than one employee on every shift for this reason. A future solution might add additional constraints that are informed by how busy every shift (day, night, evening) is. Thirdly, a high-quality solution would have fairly uniform work spread among all the employees. It wouldn't be fair if one employee got a large amount of off shifts, and another employee got very few.

Time Spent

20 hours