Modelling for Combinatorial Optimization (1DL449) Uppsala University – Autumn 2016 Course Project by Team 3

Erik Vesterlund Kristian Järvenpää

9th January 2017

In this document we present the results for our project. All experiments were run under Linux Ubuntu-14.04 (64 bit) on an Intel Xeon E5520 of 2.27 Ghz with 8 MB L2 cache and a 24 GB RAM. To measure execution time of models, we used MiniZinc's `--output-time` flag.

A Problem Description

A certain business divides its workers into teams. The work entails driving which means that every team must have at least one person who has a driver's license. Each worker naturally has favorites among his coworkers that he prefers to work with. The boss's problem is to divide his workers into teams such that each team has a driver and the workers are happy with their team mates.

A.1 Model

Let n be the number of workers and assume that all teams are of equal size $m = n/t$ where t is the number of teams, and let the number of drivers be $d \geq t$. Let P be an $n \times n$-matrix with integer entries $0 \leq p_{ij} \leq n-1$, referred to as the preference matrix. Each entry denotes worker i's strict preference for working in the same team as worker j; a value of 1 means most preferred while a value of $n-1$ means least preferred. Each person "prefers" himself the most so $p_{ii} = 0$ for all i. Note that preference is not necessarily reciprocal, meaning P need not be symmetrical.

We assume that the workers labeled 1 through d are the ones who have a driver's license. A division of the workers into teams is referred to as a shift. A shift S, then, is simply a permutation of the set $\{1, ..., n\}$. We let $S_1, ..., S_m$ be the first team, $S_{m+1}, ..., S_{2m}$ the second team, and so forth. The "first" position of each team is assigned to the driver, i.e. $S_1, S_{m+1}, ..., S_{(t-1)m+1}$ are drivers.

As a symmetry-breaking constraint we require that the selected drivers appear in order from left to right, i.e. $S_1 < S_{m+1} < \ldots$. Furthermore, there is no hierarchy within a team, so we may add a second symmetry-breaking constraint requiring that the members of each team appear in order, i.e. that $S_{\ell m+2} < S_{\ell m+3} < \ldots < S_{\ell m+m}$, $\ell = 0, 1, \ldots, t-1$.

In the problem origin, team size m was 3, which is what we have used when running the model. As for the number of drivers, the solution space is larger if $d > t$. It quickly became clear that even the smaller instances were difficult to solve (given our timeout limit) and so we only tried to find solutions for $d = t$.

A.2 Preference Instances

Letting MiniZinc generate a preference matrix leads to rather "artificial" matrices, such as

$$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}$$

A better way, in our estimation, would be to let MiniZinc generate all matrices as defined above, then choose one or more that look "random enough" and try to maximize satisfaction for those particular instances. Unfortunately this is wholly infeasible, as it takes too long before MiniZinc starts outputting the ones that look more random. Thus it is part of the model itself to generate the preference matrix.

A.3 Worker Satisfaction

Each worker's satisfaction with his team can range from $\frac{(m-1)m}{2}$ to $(m-1)(n-1)$ (lower is better). It's not straightforward how to measure overall worker satisfaction, however. Given P it's not likely, nor is it realistic, to try to enforce that each worker is maximally satisfied. A reasonable constraint is that no one has to work with the person he dislikes the most.

The final objective is to maximize the average satisfaction across the shift. To avoid maximizing a floating point value, the sum of the workers' individual levels of satisfaction was maximized, since if y is a maximal sum then y/n is a maximal average.

B Design and Evaluation

B.1 Description

B.1.1 Implementation

The described model, with the comments indicated above, is uploaded as file `projectmodel.mzn`.

Compilation and Running Instructions. Model can be compiled and run using command `mzn-backend projectmodel.mzn -d "l=x"`, where $l \geq 3$.

Same Test-Run Commands.

```
$> mzn-gecode projectmodel.mzn -d "l=3"

----------
==========
```

B.1.2 Evaluation

We have chosen the backends for Gecode, Chuffed, OscaR, Gurobi, Yices, and MinisatID. Results are presented in table 1. On trial runs only Chuffed, OscaR and MinisatID were able to produce any results with a 90 second timeout, so we discarded the other solvers and only used those three.

Technology	LCG		CBLS		Hybrid	
Solver	Chuffed		OscaR		MinisatID	
Instance	sol	time	sol	time	sol	time
3	**27**	2.55	27	TO	27	TO
5	**45**	77.36	111	TO	83	TO
7	-	TO	362	TO	322	TO
9	-	TO	-	TO	737	TO

Table 1: Results of model tested on different solvers The ‘sol’ column gives the best solution obtained by each solver as the sum of workers satisfaction, or ’-’ if no feasible solution was found before timing out or crashing. In the ‘time’ column, the time to the first solution is given in seconds, ’error’ if our model crashed or ’TO’ if solver was timed-out. Time-out was set to 90 seconds. Note that each solution in the table should be divided by the instance parameter $(3, 5, 7, 9)$ to get the objective value that was intended.

For the largest problem instance, only MinisatID was able to produce an intermediate solution, but one has to keep in mind that Yices would not provide one in any case. In conclusion, it seems that MinisatID would be the solver to go in with this model.

Since MinisatID performed the best we did a separate run using only that solver but with an extended timeout to two minutes. The results are presented in 2. As we see only slight improvements were made and, based on instance 5 in 1 we have no reason to believe that solutions for the larger instances are anywhere near an optimum.

Solver	MiniSAT	
Instance	sol	time
3	27	TO
5	72	TO
7	317	TO
9	722	TO

Table 2: Results of testing our model on cuffed with extended time-out. The ‘sol’ column gives the best solution obtained by each solver as the cost of the obtained schedule or ’-’ if no feasible solution was found before timing out or crashing. In the ‘time’ column, the time to the first solution is given in seconds, ’error’ if our model crashed or ’TO’ if solver was timed-out. Time-out was set to 120 seconds

Crash report

No errors to report.

Feedback

Intellectual Property

We certify that our report and all its uploaded attachments were produced solely by our team, except where explicitly stated otherwise and clearly referenced, that each teammate can individually explain any part starting from the moment of submitting our report, and that our report and attachments are not and will not be freely accessible on a public repository.

Collaboration was done using Overleaf for the report, and a private Github repository for the code.