Department of Mathematics and Computer Science
University of Southern Denmark, Odense

September 20, 2020
Marco Chiarandini

DM877 - Constraint Programming

Obligatory Assignment 1, Autumn 2020[pdf format]

~~**Deadline: Saturday, October 3, 2020 at noon**~~.
Deadline: Tuesday, October 6, 2020 at noon.

1. This is the first of two preparatory, obligatory assignments on constraint programming with pass/fail evaluation.
2. The aim of this first assignment is to get you have hands on experience with modeling in CP and the implementation in MiniZinc. Both skills will be tested in the final assignment that will be graded.
3. You are encouraged to search feedback and inspiration among your peers and ask questions in class related to the assignment. Working in pairs is allowed, but the final submission must be individual. You are therefore recommended to develop source code on your own and write the report individually.
4. Note: Changes to this document may happen. They will be emphasized by different colors.
5. You have to submit:
   - a PDF document containing max five pages of description of the model and the results you have obtained on the data set provided
   - source code of your implementation of the model in MiniZinc.
6. The submission is via SDU Assignment in Black Board.
7. The data set associated with the assignment is available at:
   `https://imada.sdu.dk/~marco/DM877/assets/tiny_jobs.tgz`.

1 Problem description

Shops, duties and shifts

A company has 4 shops distributed in a city. There are two types of duties, daytime duties and nighttime duties.
A *daytime duty* is characterized by an *interval* of an arbitrary range (within 7:00 and 20:00), a *location* (of the 4 shops), a *skill* that a worker should have. A *time interval* is composed by *time slots* where a slot corresponds to one hour. Each duty requires that at least one worker is *covering* it at any time during its extent. However, a duty with long duration can be *split* and covered by different workers. Each part of a split duty is called *shift*. A duty that is declared *fixed* cannot be split and constitues a shift on its own. Hence, shifts are continuous intervals of working time, or sequences of consecutive slots of work.
A *nighttime duty* has a fixed interval (from 20:00-6:00) which cannot be split, a location, but no specific skills are required (any workers can cover it). Hence, a nighttime duty is also called a *nighttime shift*. A worker typically starts a nighttime shift at 20:00 without having done any daytime shifts before or he/she can go on a nighttime shift if his/her last shift of a day ends at 20:00 and then continues during the night. In fact, during a nighttime shift, a worker can sleep, thus he/she can work longer than 10 hours. If a worker has done a nighttime shift, then the day after, he/she cannot do any daytime shift.

Requirements

Requirements are given in two data files: `demands.csv` and `demands_night.csv`, skecthed in Fig. 1. Daytime duties in `demands.csv` are characterized by:

- Skills: `SKILL1..SKILL5`
- Location: 1,2,3,4
- Day: `Monday..Sunday`
- From: on hourly basis (if not on the clock round to the largest hour smaller than the given time)
- To: on hourly basis (if not on the clock round to the largest hour smaller than the given time)
- Minimum: number of workers needed at least
- Fixed: whether the duty can be split into shorter shifts or not.

Nighttime duties in `demands_night.csv` are characterized by:

- Location: 1,2,3,4
- Day: `Monday..Sunday`
- Overnight: number of workers needed at least

Workers

Data about the workers are given in another data file: `staff.csv`, sketched in Fig. 2. Workers are characterized by:

- skills (five skills): `SKILL1..SKILL5`,
- total work hours per week: `ideal_daytime`
- minimum, maximum and ideal daytime per week excluding nighttime shift hours: `min_daytime`, `max_daytime`, `ideal_daytime`
- limits on the shift length (minimum and maximum): `min_daytime_shift`, `max_daytime_shift`
- whether nighttime shift can be taken: `overnight`
- limits on the number of nighttime shifts (minimum and maximum): `min_overnight_shift`, `max_overnight_shift`
- shop preference: `location1..location4`

- preference for morning (7-13) or afternoon (13-20) shifts: `weekday_morning_preference`, `weekday_afternoon_preference`
- availability in terms of time slots on a daytime basis where a worked is not available,
- wages (per hour): `Daytime_wage`
- happiness importance: `HappinessImportance`

The *happiness* of a worker dinimishes due to the following factors:

- allocation to a location not among the preferred ones,
- and presence of shifts outside the preferred morning and afternoon hours.
- weekly working hours discrepant from the ideal weekly amount,

In addition, the happiness of each worker has a different importance, reflecting the difference in seniority level. When aggregating the loss of workers' happinesses, worker importances are considered as weights for each worker.

Goal

The aim of the assignment is to find a weekly schedule for the workers that satisfies all the required duties, workers' constraints and at the same time maximizes the overall workers' happiness and minimize company's expense. These two objectives might be in trade off. Hence, we will adopt a lexicographic approach by defining an order between the objective and optimizing them sequentially by forbidding to worsen those previously optimized.

Constraints

- A worker can do at most three shifts per day if the shifts are not continuous
- A daytime duty must be covered by at least one worker at any time. If allowed the duty can be split among different workers.
- The workers covering a duty must (all of them) have the skill required by the duty.
- A worker cannot work for a time slot when he is not available.[1]
- A worker cannot work for a nighttime shift if it has the attribute `Overnight` to `False` and the maximum number of night shifts is 0.
- Any shift that a worker takes must have a duration within the the minimum and maximum shift length required by the worker.
- the total amount of daytime worked hours per worker in a week must be within the range of daytime hours per week Note that, night hours are excluded, because they are counted in another way (as required by the company.
- The number of night duties for a worker must be within the min and max value of the worker.
- If a worker has worked the night shift the he/she cannot work in the daytime duty of the following day.

Optimization criteria

The following criteria are to be minimized in the given order:

1. number of slots in which a worker is not working in a preferred location weighted by happiness importance,
2. number of slots outside the preferred morning and afternoon hours weighted by happiness importance,
3. number of slots of work discrepant from the ideal weekly amount weighted by happiness importance,
4. company overall expenses.

[1] If a worker is unavailable until, eg, 11pm then he/she is not available for the whole nighttime duty.

	Skills	Location	Day	From	To	Minimum	Fixed
0	SKILL4	1	Monday	7:00	14:00	1	True
1	SKILL4	1	Monday	13:45	20:00	1	True
2	SKILL4	1	Tuesday	7:00	14:00	1	True
..	...	...	...	...	...	...	...
140	SKILL1	3	Thursday	9:00	20:00	1	False
141	SKILL1	3	Friday	9:00	20:00	1	False

	Location	Day	Overnight
0	1	Monday	0
1	1	Tuesday	0
2	1	Wednesday	0
..	..	...	..
26	3	Saturday	0
27	3	Sunday	0

Figure 1: The requirements files `demands.csv` `demands_night.csv`.

2 Data

The data are organized in three files: `demands.csv`, `demands_night.csv`, `staff.csv`. The first two indicate the duty demand and the last one reports details about the worker. You find some excerpts of these files in Figures 1 and 2.

Additionally, you are given a Python script `prepare.py` to transform the data in the MiniZinc format that you prefer. The script is to be completed according to your needs consequent to your model.

Employee	1	2	3	...	37	38
HappinessImportance	7	5	7	...	7	5
SKILL1	True	True	True	...	True	True
SKILL2	True	True	False	...	True	False
SKILL3	True	False	False	...	True	False
SKILL4	True	False	False	...	True	False
SKILL5	False	False	False	...	False	False
Overnight	True	False	False	...	True	True
Daytime_wage	16	15.5	15	...	17	15.5
min_daytime	30	20	30	...	32	4
ideal_daytime	35	40	40	...	40	8
max_daytime	35	42	42	...	42	12
min_daytime_shift	6	6	6	...	4	2
max_daytime_shift	10	10	10	...	10	6
min_overnight_shift	0	0	0	...	0	3
max_overnight_shift	1	0	0	...	1	3
location1	1	0	0	...	0	0
location2	1	0	1	...	0	1
location4	1	0	-1	...	-1	-1
location3	0	0	1	...	0	-1
weekday_morning_preference	1	0	0	...	0	0
weekday_afternoon_preference	0	0	0	...	0	0
Range (12 hrs)	1pm-5pm	NaN	NaN	...	NaN	NaN
mon_start	13:00	NaN	NaN	...	NaN	NaN
mon_end	17:00	NaN	NaN	...	NaN	NaN
Range (12 hrs).1	NaN	7am-3pm	NaN	...	NaN	NaN
tue_start	NaN	7:00	NaN	...	NaN	NaN
tue_end	NaN	15:00	NaN	...	NaN	NaN
Range (12 hrs).2	1pm-5pm	NaN	NaN	...	NaN	NaN
wed_start	13:00	NaN	NaN	...	NaN	NaN
wed_end	17:00	NaN	NaN	...	NaN	NaN
Range (12 hrs).3	NaN	7am-3pm	NaN	...	NaN	NaN
thu_start	NaN	7:00	NaN	...	NaN	NaN
thu_end	NaN	15:00	NaN	...	NaN	NaN
Range (12 hrs).4	NaN	NaN	NaN	...	NaN	NaN
fri_start	NaN	NaN	NaN	...	NaN	NaN
fri_end	NaN	NaN	NaN	...	NaN	NaN
Range (12 hrs).5	NaN	NaN	NaN	...	6pm-11pm	NaN
sat_start	NaN	NaN	NaN	...	18:00	NaN
sat_end	NaN	NaN	NaN	...	23:00	NaN
Range (12 hrs).6	NaN	NaN	NaN	...	6pm-11pm	NaN
sun_start	NaN	NaN	NaN	...	18:00	NaN
sun_end	NaN	NaN	NaN	...	23:00	NaN

Figure 2: The workers file `staff.csv`.

3 Remarks

- Start small (few data and few constraints) and work incrementally. You do not need to address all constraints to pass the assignment.