UofT CARTE Labatt ML Bootcamp

Lab 3

Lab author: Alexander Olson, alex.olson@utoronto.ca

In this lab we will look at practical examples of using optimization, with the open source Google OR-Tools toolkit. Let's install that now:

```
In [1]: !pip install -U ortools
```

```
Requirement already satisfied: ortools in /Users/alex/miniconda3/envs/bootcamp/lib/pytho
n3.9/site-packages (9.3.10497)
Requirement already satisfied: absl-py>=0.13 in /Users/alex/miniconda3/envs/bootcamp/li
b/python3.9/site-packages (from ortools) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in /Users/alex/miniconda3/envs/bootcamp/li
b/python3.9/site-packages (from ortools) (1.22.4)
Requirement already satisfied: protobuf>=3.19.4 in /Users/alex/miniconda3/envs/bootcamp/
lib/python3.9/site-packages (from ortools) (4.21.1)
```

In this lab, we are going to work through a scheduling problem which is slightly more complex than the one in the lecture.

Scheduling

Organizations with employees that work numerous shifts must schedule enough people for each shift. The schedules will usually include restrictions such as "no employee should work two shifts in a row." Finding a schedule that meets all of the requirements might be time consuming.

In the following scenario, a hospital supervisor must construct a timetable for four nurses over the course of three days, subject to the following constraints:

- Each day is divided into three shifts of eight hours each.
- Every day, a single nurse is assigned to each shift, and no nurse works more than one shift at a time.
- During the three-day period, each nurse is allocated to at least two shifts.

Let's import the model and get started:

```
In [2]: from ortools.sat.python import cp_model
```

YOUR TURN

We start by describing the parameters of our problem programatically. In the following cell, create variables `all_shifts`, and `all_days` which list the possible values for these attributes. `all_nurses` is done for you.

```
In [3]: all_nurses = [0,1,2,3]
        all_shifts = [0,1,2]
        all_days = [0,1,2]
```

In [4]:
```python
num_nurses = len(all_nurses)
num_shifts = len(all_shifts)
num_days = len(all_days)
```

We are now going to initialise our model, and create a variable for every possible combination of nurse, shift and day. In the lecture example, this was our variable x_i^j which was equal to 1 if task i was assigned to machine j. We are doing the same here for the nurses, except we have three variables: nurse, day, and shift.

We do this by looping through all three of our variables in sequence and adding that possibility to a dictionary. This tells the model that we have a set of boolean (zero or one) variables that it will need to consider:

In [5]:
```python
model = cp_model.CpModel() #Create the empty model
shifts = {} #We are going to save references to our variables here to use later
for n in all_nurses:
    for d in all_days:
        for s in all_shifts:
            shifts[(n, d,
                    s)] = model.NewBoolVar('shift_n%id%is%i' % (n, d, s))

print(f'Created {len(shifts)} hypothetical shift assignments')
```

```
Created 36 hypothetical shift assignments
```

Now that we have our possible options, we need to start defining what we want the model to do with them.

As we said, each nurse can only work one shift at a time, and only one shift per day. We will tell the model that for each shift on each day, exactly one assignment should have a value of 1:

In [6]:
```python
for d in all_days:
    for s in all_shifts:
        model.AddExactlyOne(shifts[(n, d, s)] for n in all_nurses) #This says that for e
```

YOUR TURN

Next, we need to say that on each day, a nurse should get at most one shift. The function to do this is `model.AddAtMostOne()`, and it works the same as the function above. In the cell below, fill out code to tell the model that for each nurse on each day, at most a single hypothetical assignment should be 1.

In [7]:
```python
for n in all_nurses:
    for d in all_days:
        model.AddAtMostOne(shifts[(n,d,s)] for s in all_shifts)
```

In our scenario, there are nine total shifts over the three-day period. With three nurses, that means that two nurses will work two shifts, and an unlucky third nurse will work three shifts. We need to tell the model that every nurse should work between 2 and 3 shifts.

In [8]:
```python
min_shifts_per_nurse = (num_shifts * num_days) // num_nurses #Divide the shifts between
print(f'Minimum shifts per nurse: {min_shifts_per_nurse}')
```

```
Minimum shifts per nurse: 2
```

YOUR TURN

Now we will define the upper bound. In the box below, write code that checks whether the total number of shifts can be cleanly divided between the nurses. If it can, the maximum is the same value as the minimum - set this in a variable called `max_shifts_per_nurse`.

If there is a leftover shift, then the maximum will be one more than the minimum.

In [9]:
```
import math

true_shifts = (num_shifts * num_days) / num_nurses

min_shifts_per_nurse = math.floor(true_shifts)

max_shifts_per_nurse = math.ceil(true_shifts)

print(f'Max shifts per nurse: {max_shifts_per_nurse}')
print(f'Min shifts per nurse: {min_shifts_per_nurse}')
```

```
Max shifts per nurse: 3
Min shifts per nurse: 2
```

We are almost finished defining our problem! We will now tell the model how to keep track of the number of shifts worked by each nurse, so that we can then add our constraints on the max and min number of shifts:

In [10]:
```
for n in all_nurses:
    #We create a variable which stores all of the potential shifts this nurse
    #could have worked, so that the model knows how to count it
    num_shifts_worked = []
    for d in all_days:
        for s in all_shifts:
            num_shifts_worked.append(shifts[(n, d, s)])

    #Then we tell the model that the total value for that
    #nurse must be between the maximum and the minimum
    model.Add(min_shifts_per_nurse <= sum(num_shifts_worked))
    model.Add(sum(num_shifts_worked) <= max_shifts_per_nurse)
```

Now we initialize our solver, and tell it that it can give us every possible solution (which is fine since this is a small example).

In [11]:
```
solver = cp_model.CpSolver()
solver.parameters.linearization_level = 0 #Tells the model we don't need to relax the in
# Enumerate all solutions.
solver.parameters.enumerate_all_solutions = True
```

The last component before we call our solver is some handy (and optional) code which will print out the results as the solver works.

In [12]:
```
class NursesPartialSolutionPrinter(cp_model.CpSolverSolutionCallback):
    """Print intermediate solutions."""

    def __init__(self, shifts, num_nurses, num_days, num_shifts, limit):
        cp_model.CpSolverSolutionCallback.__init__(self)
        self._shifts = shifts
        self._num_nurses = num_nurses
        self._num_days = num_days
        self._num_shifts = num_shifts
        self._solution_count = 0
        self._solution_limit = limit
    def on_solution_callback(self):
        self._solution_count += 1
        print('Solution %i' % self._solution_count)
        for d in range(self._num_days):
            print('\tDay %i' % d)
            for n in range(self._num_nurses):
                is_working = False
                for s in range(self._num_shifts):
                    if self.Value(self._shifts[(n, d, s)]):
                        is_working = True
                        print('\t\tNurse %i works shift %i' % (n, s))
                if not is_working:
                    print('\t\tNurse {} does not work'.format(n))
        print()
        if self._solution_count >= self._solution_limit:
            print('Stop search after %i solutions' % self._solution_limit)
            self.StopSearch()

    def solution_count(self):
        return self._solution_count

# Display the first five solutions.
solution_limit = 5
solution_printer = NursesPartialSolutionPrinter(shifts, num_nurses,
                                                num_days, num_shifts,
                                                solution_limit)
```

And now we are ready to run our solver!

```
In [13]: solver.Solve(model, solution_printer)
```

```
Solution 1
        Day 0
                Nurse 0 does not work
                Nurse 1 works shift 0
                Nurse 2 works shift 1
                Nurse 3 works shift 2
        Day 1
                Nurse 0 works shift 2
                Nurse 1 does not work
                Nurse 2 works shift 1
                Nurse 3 works shift 0
        Day 2
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 works shift 0
                Nurse 3 does not work

Solution 2
        Day 0
                Nurse 0 works shift 0
                Nurse 1 does not work
                Nurse 2 works shift 1
                Nurse 3 works shift 2
        Day 1
                Nurse 0 does not work
                Nurse 1 works shift 2
                Nurse 2 works shift 1
                Nurse 3 works shift 0
        Day 2
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 works shift 0
                Nurse 3 does not work
```

```
Solution 3
        Day 0
                Nurse 0 works shift 0
                Nurse 1 does not work
                Nurse 2 works shift 1
                Nurse 3 works shift 2
        Day 1
                Nurse 0 works shift 1
                Nurse 1 works shift 2
                Nurse 2 does not work
                Nurse 3 works shift 0
        Day 2
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 works shift 0
                Nurse 3 does not work

Solution 4
        Day 0
                Nurse 0 works shift 0
                Nurse 1 does not work
                Nurse 2 works shift 1
                Nurse 3 works shift 2
        Day 1
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 does not work
                Nurse 3 works shift 0
        Day 2
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 works shift 0
                Nurse 3 does not work

Solution 5
        Day 0
                Nurse 0 does not work
                Nurse 1 works shift 0
                Nurse 2 works shift 1
                Nurse 3 works shift 2
        Day 1
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 does not work
                Nurse 3 works shift 0
        Day 2
                Nurse 0 works shift 2
                Nurse 1 works shift 1
                Nurse 2 works shift 0
                Nurse 3 does not work

Stop search after 5 solutions
```

```
Out[13]: 2
```

As you can see, the model very quickly provides us with a whole range of options that fit our criteria! This is super speedy because the example we've given is so small, but it demonstrates how a task which would be very complex to handle manually can be quickly solved using linear programming.

BONUS (Optional)

1. We only consider our schedule over three days, for a small set of nurses. Can you go back and add a wider range for the model to consider, with more staff?

2. Our model is currently happy to assign a nurse to work every single day. Can you add a restriction that nurses must get at least one day off? How about more?
3. A model is `feasible` if there is any solution which satisfies all the criteria. Can you add a requirement that makes this problem infeasible?
4. **CHALLENGE**: Can you define shifts of different lengths, and then rewrite the existing restrictions to refer to the total number of hours worked? *Hint*: try creating separate variables for half- and full-length shifts.