scheduling_analysis_example

July 26, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
from IPython.display import Image
import seaborn as sns
```

[2]:
```
%matplotlib inline
```

1 A Basic Tactical Scheduling Analysis Example

The multiweek tour scheduling model (MWTS) was developed to use for *tactical scheduling analysis* problems. The focus of such problems is on evaluation and comparison of different staff scheduling policies and practices. Metrics for comparison might include overall staffing costs, understaffing levels, and schedule quality. Tactical scheduling models are not really intended for use in the ongoing process of creating tour schedules for a fixed cohort of staff - a process we call *operational scheduling*. However, tactical scheduling models such as MWTS do indeed produce actual multiweek tour schedules. These schedules supplement broader model output metrics such as staffing costs by showing concrete examples of how a given scheduling policy might actually be implemented in practice. So, let's see an example of a basic tactical scheduling analysis problem in which we evaluate the relative merits of various mixes of full and part-time tours and use of multiple shift lengths.

This notebook is **not** intended to be a in-depth description of the MWTS model. For that, see our paper preprint at - LINK TO PAPER. Instead, we are just showing how such a model gets used in practice. Single week versions of this model were used in numerous real scheduling analysis projects and the technical details of that model can be found in this earlier paper:

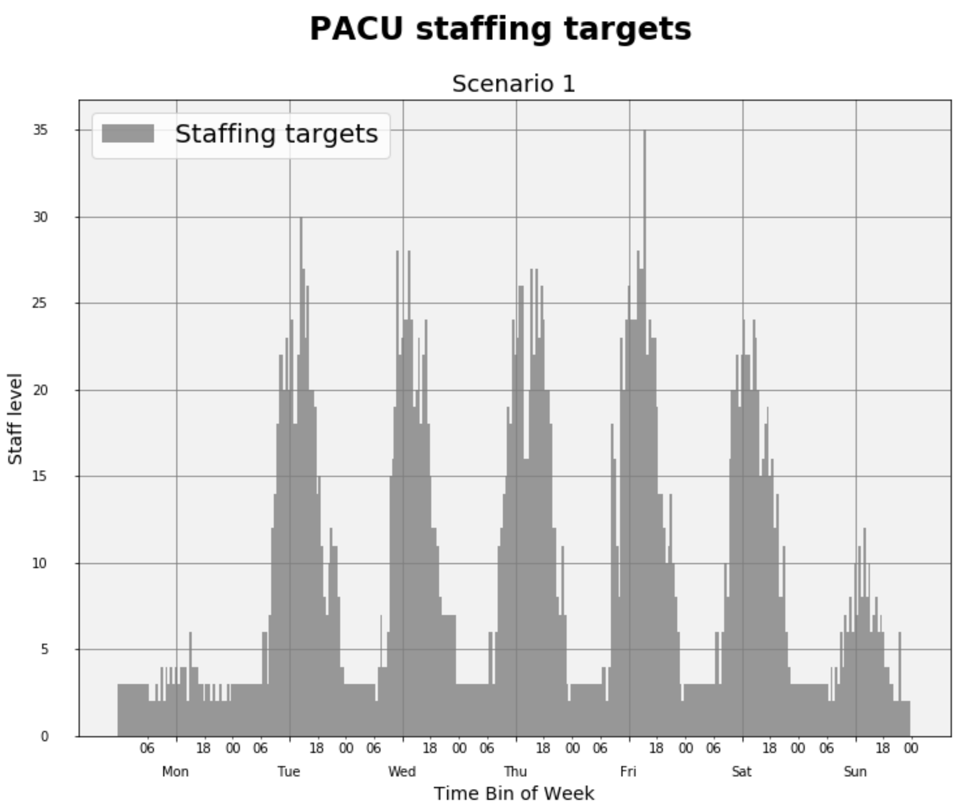
Isken, Mark W. "An implicit tour scheduling model with applications in healthcare." *Annals of Operations Research* 128.1-4 (2004): 91-109.

1.1 The PACU staffing targets

For this example, we will use staffing requirements from a hospital post-anesthesia care unit (PACU). Let's assume that staffing level targets by half-hour for each day of the week have already been set from some previous analysis. As you can see from the following plot, PACU staffing targets exhibit signficant time of day and day of week effects.

[3]:
```python
# Plot of PACU demand
Image(filename="pacu_staffing_targets.png")
```

[3]:



1.2 Tour types

Currently our PACU is staffed with full-time nurses who each work five, eight-hour shifts, each week. Let's call this *tour type 1*, or TT1 for short. We would like to consider using other tour types such as part-time tours or the use of ten-hour shifts. For this example, we are going to look at combinations of the following three tour types:

- **TT1** - full-time, five eight-hour shifts worked each week
- **TT2** - part-time, five eight-hour shifts worked **every other** week (with a minimum of two and a maximum of three shifts worked each week)
- **TT3** - full-time, four ten-hour shifts worked each week

We are curious if the use of TT2 and/or TT3 will allow us to better meet the variable staffing targets of our PACU.

1.3 Other scheduling policy parameters

We are only varying the mix of tour types in this example. Other scheduling inputs are held fixed across the different scenarios. These inputs include:

- **Scheduling horizon** - Four weeks, each day is made up of forty-eight half-hour periods
- **Allowable shift start times** - Shifts can start on any half hour of the day except those resulting in a shift ending between midnight and 5am. Within each tour, each shift starts at the same time each day.
- **Staffing costs** - Just using a sum of number of periods scheduled. The model can handle different tour type specific costs but we won't consider that here.
- **Understaffing costs** - We put relatively high penalties on understaffing.
- **Weekend policies** - People can work a maximum of four weekend days over four weeks and maximum of two weekends in which at least one of the days is worked.

1.4 Scenario analysis

Let's look at a few different combinations of allowable tour types and see how they compare in terms of total staffing costs as well as in the amount of understaffing. For each scenario we have created a data input file in the well known AMPL compatible DAT format. We will generate and solve the MWTS model for each scenario and compare the outputs.

1.4.1 Technical preamble

If you want to get this notebook and associated data files and try this for yourself, you'll need to do a few things. This notebook assumes you are already comfortable with using Python (the Anaconda distribution is recommended) and familiar with optimization software (e.g. CBC, glpk, or Gurobi), git, Github, installing Python programs using pip, Python virtual environments, and running programs from a command shell. I use Ubuntu Linux and this example is Linux based. Of course you can just read through the notebook to get the gist of it.

- Need to have either CBC, glpk or Gurobi installed and available to use as the mixed-integer programming solver
- Clone or download the source code from https://github.com/misken/pymwts
- Open a command shell in the main project director pymtws/.

It is recommended to create a virtual environment within which to install pymwts to avoid adding such tools to your base Python environment. Then just use pip to install it and navigate to the examples/ subfolder after installation is complete. The pymwts package depends on a few other Python packages, namely, pandas and pyomo. Both of these will get installed automatically if they aren't already installed.

- pip install .
- cd examples

This notebook and the data files are in this examples/ folder and the examples/input/ subfolder, respectively.

After installing pymwts, you can run it from a command shell. Let's run it with the -h flag to get the help info about pymwts.

[4]:
```
!pymwts -h
```

```
usage: pymwts [-h] [-p PATH] [-s {cbc,glpk,gurobi}] [-t TIMELIMIT] [-g MIPGAP]
              [--version]
              scenario phase1dat

Solve a multi-week tour scheduling problem.

positional arguments:
  scenario              Short string to be used in output filenames
  phase1dat             DAT file for phase 1

optional arguments:
  -h, --help            show this help message and exit
  -p PATH, --path PATH  Relative path to output file directory. Terminate with
                        /
  -s {cbc,glpk,gurobi}, --solver {cbc,glpk,gurobi}
                        cbc, glpk or gurobi for now
  -t TIMELIMIT, --timelimit TIMELIMIT
                        seconds
  -g MIPGAP, --mipgap MIPGAP
                        Can prevent really long run times.
  --version             show program's version number and exit

May the force be with you.
```

[5]:
```
!pymwts --version
```

```
pymwts 0.1.0
```

1.4.2 Big picture of the solution process

TODO: Diagram of models and data files and solvers and such…

1.4.3 Scenario 1 - TT1 only

In this first scenario we will only use TT1 tour types (full-time, five eight-hour shifts per week). This scenario represents the case of the least scheduling flexibility that we'll consider in this analysis. The AMPL data file is named scenario1_tt1.dat and is in the input/ subdirectory. The scenario name will be scenario1_tt1. We'll set a timelimit of 600 seconds and set the mipgap to 2%. We will specify that the output files should get written to the output/ subdirectory. I'm using the Gurobi solver (academic edition). When we run this, we'll see a bunch of output generated by Pyomo, by the solver, and by various pieces of the pymwts package. In addition, numerous output files are generated and we'll be using some of these in our analysis. We'll know that everything solved correctly if the last bit of output to the screen is `Output files created'.

[6]:
```
!pymwts scenario1_tt1 ./input/scenario1_tt1.dat -s gurobi -p ./output/ -t 600␣
↪-g 0.02
```

```
Namespace(mipgap=0.02, path='./output/', phase1dat='./input/scenario1_tt1.dat',
scenario='scenario1_tt1', solver='gurobi', timelimit=600)

*** Scenario scenario1_tt1


*** Phase 1 model instance created.


*** Setting up the solver.


*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmp0kjy1ksk.pyomo.lp
Reading time = 0.06 seconds
x13672: 16024 rows, 13672 columns, 141709 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 16024 rows, 13672 columns and 141709 nonzeros
Variable types: 9409 continuous, 4263 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [6e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 10597 rows and 8996 columns
```

```
Presolve time: 0.14s
Presolved: 5427 rows, 4676 columns, 33017 nonzeros
Variable types: 1380 continuous, 3296 integer (416 binary)

Root relaxation: objective 1.641319e+04, 5708 iterations, 0.82 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 16413.1892    0  951          - 16413.1892      -     -    1s
     0     0 16413.8824    0  735          - 16413.8824      -     -    1s
     0     0 16413.9241    0  801          - 16413.9241      -     -    1s
     0     0 16413.9429    0  835          - 16413.9429      -     -    1s
     0     0 16413.9623    0  843          - 16413.9623      -     -    2s
H    0     0                    130928.00000 16413.9623  87.5%     -    2s
     0     0 16413.9623    0  734 130928.000 16413.9623  87.5%     -    2s
H    0     0                    16464.000000 16413.9623  0.30%     -    3s

Cutting planes:
  Gomory: 5

Explored 1 nodes (7337 simplex iterations) in 3.01 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 16464 130928

Optimal solution found (tolerance 2.00e-02)
Best objective 1.646400000000e+04, best bound 1.641396226313e+04, gap 0.3039%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpbxm1d489.pyomo.lp
Reading time = 0.02 seconds
x4321: 2785 rows, 4321 columns, 24481 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
```

```
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 2785 rows, 4321 columns and 24481 nonzeros
Variable types: 1 continuous, 4320 integer (4320 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 2785 rows and 4321 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 8 available processors)

Solution count 1: 960

Optimal solution found (tolerance 2.00e-02)
Best objective 9.600000000000e+02, best bound 9.600000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

Scenario 1 summary A number of output files get created by pymwts. Here's a listing:

[7]:
```
!ls output/scenario1*
```

```
output/scenario1_tt1.log
output/scenario1_tt1_phase1_capsum.csv
output/scenario1_tt1_phase1_results.yml
output/scenario1_tt1_phase1_shiftsum.csv
output/scenario1_tt1_phase1_summary.txt
output/scenario1_tt1_phase1_tourskeleton.csv
output/scenario1_tt1_phase2.dat
output/scenario1_tt1_phase2_ftesum.csv
output/scenario1_tt1_phase2_mwt.csv
output/scenario1_tt1_phase2_results.yml
output/scenario1_tt1_phase2_summary.txt
output/scenario1_tt1_phase2_toursum.csv
output/scenario1_tt1_phase2_tourtypesum.csv
output/scenario1_tt1_phase2_tur.csv
output/scenario1_tt1.tur
```

Let's look at the FTE (full time equivalent) summary:

```
[8]: ftesum_1 = pd.read_csv('output/scenario1_tt1_phase2_ftesum.csv')
     ftesum_1
```

```
[8]:    num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
     0         48        15360         960     7680.0      48.0    12800

        sched_eff  tot_periods_us       scenario
     0   0.833333           168.0  scenario1_tt1
```

A total of 48 tours were created. Since each tour is a full-time person, we see that this scenario results in a total of 48.0 FTEs. There's a similar output file that includes the same measures broken down by tour type. Obviously, this will be more useful in subsequent scenarios in which we use multiple tour types.

```
[9]: tourtypesum_1 = pd.read_csv('output/scenario1_tt1_phase2_tourtypesum.csv')
     tourtypesum_1
```

```
[9]:    tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
     0         1         48        15360         960     7680.0      48.0

             scenario
     0  scenario1_tt1
```

Let's also show a plot of scheduled capacity superimposed on the underlying staffing targets (for one week).

```
[10]: # Plot of cap and demand
      capacity1_df = pd.read_csv('output/scenario1_tt1_phase1_capsum.csv')
      capacity1_df = capacity1_df.loc[capacity1_df['week'] == 1]
      capacity1_df = capacity1_df.sort_values(by=['day', 'period'])
      capacity1_df
```

```
[10]:       period  day  week  dmd  cap  us1  us2  ustot
      0          1    1     1    3  2.0  1.0  0.0    1.0
      28         2    1     1    3  2.0  1.0  0.0    1.0
      56         3    1     1    3  2.0  1.0  0.0    1.0
      84         4    1     1    3  2.0  1.0  0.0    1.0
      112        5    1     1    3  2.0  1.0  0.0    1.0
      …        …  …    …  …    …  …    …     …
      1228      44    7     1    6  5.0  1.0  0.0    1.0
      1256      45    7     1    2  4.0 -0.0  0.0    0.0
      1284      46    7     1    2  3.0  0.0  0.0    0.0
      1312      47    7     1    2  3.0 -0.0  0.0    0.0
      1340      48    7     1    2  2.0  0.0  0.0    0.0

      [336 rows x 8 columns]
```

Since we will want to do one plot per scenario, we'll create a plotting function that we can reuse.

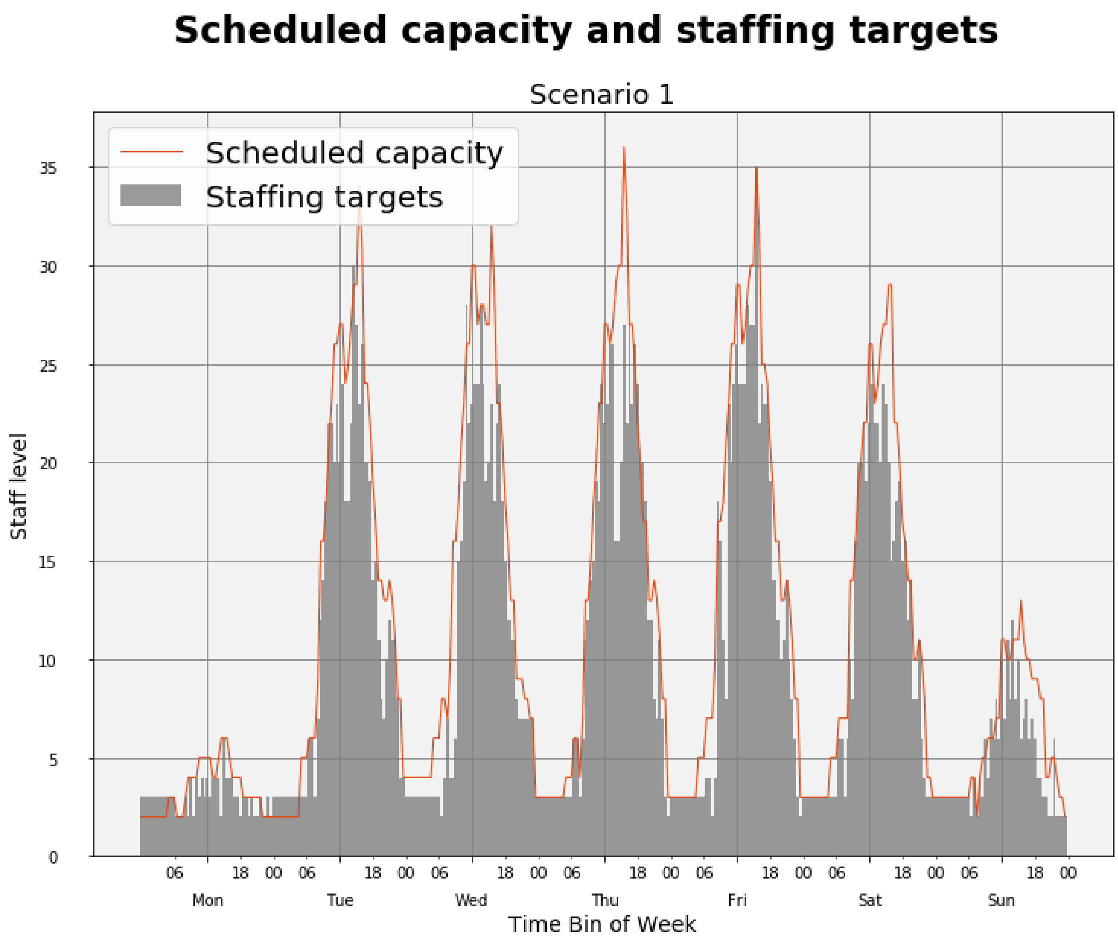
[11]:
```python
def capacity_plot(capacity_df, scenario_title, ax):

    # Create a list to use as the X-axis values
    #--------------------------------------------

    timestamps = pd.date_range('01/05/2015', periods=336, freq='30Min')

    major_tick_locations = pd.date_range('01/05/2015 12:00:00', periods=7,␣
↪freq='24H').tolist()
    minor_tick_locations = pd.date_range('01/05/2015 06:00:00', periods=28,␣
↪freq='6H').tolist()

    # Specify the mean occupancy and percentile values
    #-----------------------------------------------------------
    staffing_target = capacity_df['dmd']
    capacity = capacity_df['cap']

    # Styling of bars, lines, plot area
    #-----------------------------------

    # Style the bars for staffing targets
    bar_color = 'grey'
    bar_opacity = 0.8

    # Style the line for the scheduled capacity
    cap_line_style = '-'
    cap_color = '#dd4814'
    cap_line_width = 1

    # Set the background color of the plot. Argument is a string float in
    # (0,1) representing greyscale (0=black, 1=white)
    ax.patch.set_facecolor('0.95')

    # Can also use color names or hex color codes
    # ax2.patch.set_facecolor('yellow')
    # ax2.patch.set_facecolor('#FFFFAD')


    # Add data to the plot
    #--------------------------

    # Staffing targets as bars
    ax.bar(timestamps.values, staffing_target, color=bar_color,␣
↪alpha=bar_opacity, label='Staffing targets', width=1/48)
```

```python
    # Scheduled capacity
    ax.plot(timestamps.values, capacity, linestyle=cap_line_style,␣
↪linewidth=cap_line_width, color=cap_color, \
            label='Scheduled capacity')


    # Create formatter variables
    dayofweek_formatter = DateFormatter('%a')
    qtrday_formatter = DateFormatter('%H')

    # Set the tick locations for the axes object

    ax.set_xticks(major_tick_locations)
    ax.set_xticks(minor_tick_locations, minor=True)

    # Format the tick labels
    ax.xaxis.set_major_formatter(dayofweek_formatter)
    ax.xaxis.set_minor_formatter(qtrday_formatter)

    # Slide the major tick labels underneath the default location by 20 points
    ax.tick_params(which='major', pad=20)

    # Add other chart elements
    #-------------------------

    # Set plot and axis titles
    ax.set_title(scenario_title, fontsize=18)
    ax.set_xlabel('Time Bin of Week', fontsize=14)
    ax.set_ylabel('Staff level', fontsize=14)

    # Gridlines
    ax.grid(True, color='0.5')

    # Legend
    leg = ax.legend(loc='best', frameon=True, fontsize=20)
    leg.get_frame().set_facecolor('white')

    return ax
```

Now we can call our function to create the capacity plot for Scenario 1.

[12]:
```python
# Create a Figure and Axes object and call plot function
fig1 = plt.figure()
fig1.set_size_inches(12, 9)
fig1.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
↪fontweight='bold')
```

```
ax1 = fig1.add_subplot(1,1,1)

capacity_plot(capacity1_df, 'Scenario 1', ax1);
```



Note that there's quite a bit of overstaffing due to the lack of flexibility of only being allowed to use full-time, eight-hour tours. Now, let's move on to the next scenario.

1.4.4 Scenario 2 - TT1 and TT2

In addition to TT1, we now allow part-time staff through TT2. A limit on the total percentage of scheduled hours attributable to TT2 was set to 40%. As part-time tour types provide additional flexibility in meeting highly variable demand patterns, we often end up with solutions using almost all part-time tours, which might not be practical nor desirable.

[13]:

```
!pymwts scenario2_tt12 ./input/scenario2_tt12.dat -s gurobi -p ./output/ -t 600␣
↪-g 0.02
```

```
Namespace(mipgap=0.02, path='./output/', phase1dat='./input/scenario2_tt12.dat',
scenario='scenario2_tt12', solver='gurobi', timelimit=600)

*** Scenario scenario2_tt12


*** Phase 1 model instance created.


*** Setting up the solver.


*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmp_mxxkzqj.pyomo.lp
Reading time = 0.15 seconds
x18834: 22639 rows, 18834 columns, 362863 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 22639 rows, 18834 columns and 362863 nonzeros
Variable types: 9409 continuous, 9425 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [6e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 14187 rows and 12005 columns
Presolve time: 0.74s
Presolved: 8452 rows, 6829 columns, 202567 nonzeros
Variable types: 1828 continuous, 5001 integer (192 binary)

Root relaxation: objective 1.508607e+04, 12750 iterations, 3.96 seconds
Total elapsed time = 5.21s

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
```

```
     0     0 15086.0670    0  851          - 15086.0670      -      -    5s
H    0     0                    707308.00000 15086.0670  97.9%     -    6s
     0     0 15086.0670    0  848 707308.000 15086.0670  97.9%     -   10s
H    0     0                    15390.000000 15086.0670  1.97%     -   13s

Explored 1 nodes (15115 simplex iterations) in 13.37 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 15390 707308

Optimal solution found (tolerance 2.00e-02)
Best objective 1.539000000000e+04, best bound 1.508606698945e+04, gap 1.9749%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpe7imvlju.pyomo.lp
Reading time = 0.02 seconds
x6998: 3771 rows, 6998 columns, 38886 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3771 rows, 6998 columns and 38886 nonzeros
Variable types: 1 continuous, 6997 integer (6997 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3771 rows and 6998 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 8 available processors)
```

```
Solution count 1: 930

Optimal solution found (tolerance 2.00e-02)
Best objective 9.300000000000e+02, best bound 9.300000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

Scenario 2 summary

[14]:
```
ftesum_2 = pd.read_csv('output/scenario2_tt12_phase2_ftesum.csv')
ftesum_2
```

[14]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         65        14880         930     7440.0      46.5    12800

   sched_eff  tot_periods_us        scenario
0   0.860215            71.0  scenario2_tt12
```

[15]:
```
tourtypesum_2 = pd.read_csv('output/scenario2_tt12_phase2_tourtypesum.csv')
tourtypesum_2
```

[15]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1         28         8960         560     4480.0      28.0
1         2         37         5920         370     2960.0      18.5

         scenario
0  scenario2_tt12
1  scenario2_tt12
```

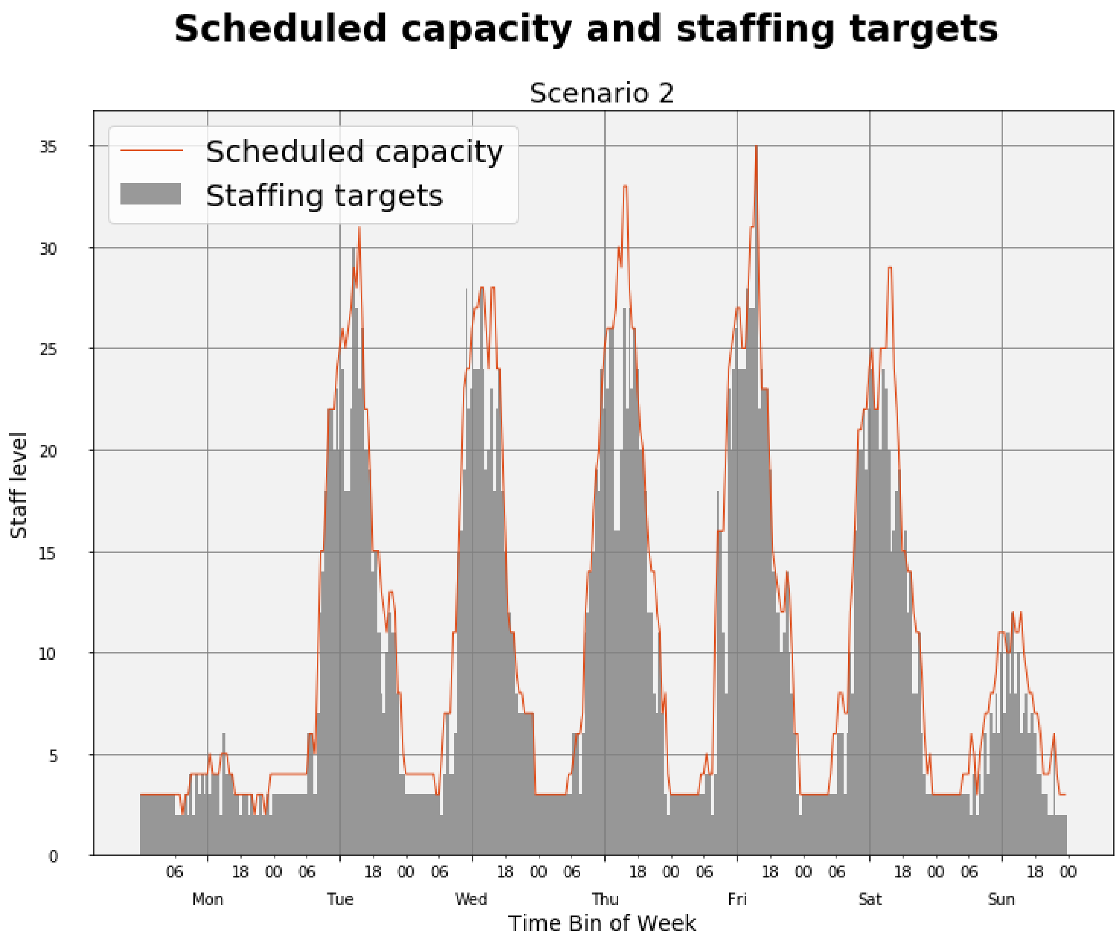
The ability to use part-time tours resulted in a savings of 1.5 FTEs. Note that the maximum level (40%) of part-time staff was used in the solution.

Create the capacity plot.

[18]:
```
# Plot of cap and demand
capacity2_df = pd.read_csv('output/scenario2_tt12_phase1_capsum.csv')
capacity2_df = capacity2_df.loc[capacity2_df['week'] == 1]
capacity2_df = capacity2_df.sort_values(by=['day', 'period'])
```

[19]:
```
# Create a Figure and Axes object and call plot function
fig2 = plt.figure()
fig2.set_size_inches(12, 9)
```

```
fig2.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
 ↪fontweight='bold')
ax2 = fig2.add_subplot(1,1,1)

capacity_plot(capacity2_df, 'Scenario 2', ax2);
```



Comparing this plot to the first plot, we see a large reduction in overstaffing. Actually, understaffing appears to be significantly reduced as well. The addition of part-time tours has helped us match the highly variable PACU demand profile.

1.4.5 Scenario 3 - TT1, TT2, and TT3

Now let's add a full-time ten-hour tour type to the mix. Perhaps the additional flexibility of a second shift length will let us match the demand variability more closely.

[20]:
```
!pymwts scenario3_tt123 ./input/scenario3_tt123.dat -s gurobi -p ./output/ -t
↪600 -g 0.02
```

```
Namespace(mipgap=0.02, path='./output/',
phase1dat='./input/scenario3_tt123.dat', scenario='scenario3_tt123',
solver='gurobi', timelimit=600)

*** Scenario scenario3_tt123


*** Phase 1 model instance created.


*** Setting up the solver.


*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmp7zp3hzac.pyomo.lp
Reading time = 0.19 seconds
x23097: 29253 rows, 23097 columns, 491159 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 29253 rows, 23097 columns and 491159 nonzeros
Variable types: 9409 continuous, 13688 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [6e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 16803 rows and 14040 columns
Presolve time: 0.89s
Presolved: 12450 rows, 9057 columns, 232016 nonzeros
Variable types: 1476 continuous, 7581 integer (368 binary)

Root simplex log…

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   12952    1.4585883e+04   6.519507e+03   0.000000e+00      5s
```

```
   18186    1.4625005e+04   0.000000e+00   0.000000e+00      7s

Root relaxation: objective 1.462500e+04, 18186 iterations, 6.20 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 14625.0045    0 1497          - 14625.0045      -     -    7s
H    0     0                    130888.00000 14625.0055  88.8%     -    9s
     0     0 14625.0055    0 1480 130888.000 14625.0055  88.8%     -   10s
     0     2 14625.0055    0 1478 130888.000 14625.0055  88.8%     -   16s
    19    22 14626.0497    5 1492 130888.000 14625.8043  88.8%   475   20s
    46    50 14627.8102    8 1382 130888.000 14625.8043  88.8%   531   25s
    62    62 14626.6468   10 1545 130888.000 14625.8043  88.8%   531   30s
    73    77 14626.6468   11 1544 130888.000 14625.8043  88.8%   612   35s
   106   102 14627.4600   18 1403 130888.000 14625.8043  88.8%   624   42s
   149   149 14628.7527   22 1370 130888.000 14625.8043  88.8%   696   50s
   198   199 14633.2967   27 1445 130888.000 14625.8043  88.8%   693   63s
   218   221 14637.4706   29 1392 130888.000 14625.8043  88.8%   698   75s
H  236   234                    18114.000000 14625.8043  19.3%   714   75s
   264   265 14637.3718   38 1395 18114.0000 14625.8043  19.3%   764   86s
H  286   265                    17528.000000 14625.8043  16.6%   736   86s
H  328   329                    17510.000000 14625.8043  16.5%   741  101s
H  348   348                    17112.000000 14625.8043  14.5%   729  101s
   390   393 14647.6923   69 1381 17112.0000 14625.8043  14.5%   740  117s
H  446   443                    17100.000000 14625.8043  14.5%   749  134s
H  458   456                    17082.000000 14625.8043  14.4%   756  134s
H  493   492                    16998.000000 14625.8043  14.0%   753  134s
   510   511 14655.1425   84 1351 16998.0000 14625.8043  14.0%   751  149s
H  559   549                    16992.000000 14625.8043  13.9%   740  149s
H  579   570                    16980.000000 14625.8043  13.9%   736  149s
   589   594 14660.1380   97 1396 16980.0000 14625.8043  13.9%   734  166s
H  621   608                    16920.000000 14625.8043  13.6%   737  166s
   666   668 14668.8269  106 1277 16920.0000 14625.8043  13.6%   734  184s
H  673   668                    16896.000000 14625.8043  13.4%   733  184s
   742   740 14676.4440  114 1236 16896.0000 14625.8043  13.4%   736  200s
H  763   755                    16884.000000 14625.8043  13.4%   738  200s
H  786   778                    16842.000000 14625.8043  13.2%   742  200s
   824   825 14684.0859  124 1171 16842.0000 14625.8043  13.2%   735  215s
H  826   825                    16818.000000 14625.8043  13.0%   735  215s
H  843   843                    15300.000000 14625.8043  4.41%   738  215s
   900   902 14695.9476  134 1373 15300.0000 14625.8043  4.41%   739  231s
H  910   902                    15288.000000 14625.8043  4.33%   736  231s
H  975   966                    15156.000000 14625.8043  3.50%   730  231s
   984   985 14704.7328  150 1226 15156.0000 14625.8043  3.50%   728  249s
H 1058  1041                    15150.000000 14625.8043  3.46%   720  249s
H 1077  1061                    15032.000000 14625.8043  2.70%   719  249s
  1079  1076 14724.2742  171 1020 15032.0000 14625.8043  2.70%   721  265s
```

```
H 1104  1076                     15026.000000 14625.8043  2.66%   715  265s
H 1154  1127                     14942.000000 14625.8043  2.12%   699  265s
  1204  1207 14733.0725  213  929 14942.0000 14625.8043  2.12%   702  280s
H 1230  1207                     14930.000000 14625.8043  2.04%   700  280s
  1344  1344 14744.4494  242  749 14930.0000 14625.8043  2.04%   679  294s
H 1434  1408                     14918.000000 14625.8043  1.96%   653  294s

Explored 1558 nodes (1012770 simplex iterations) in 294.48 seconds
Thread count was 8 (of 8 available processors)

Solution count 10: 14918 14930 14942 … 16818

Optimal solution found (tolerance 2.00e-02)
Best objective 1.491800000000e+04, best bound 1.462580434361e+04, gap 1.9587%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpxppx763d.pyomo.lp
Reading time = 0.02 seconds
x6877: 3713 rows, 6877 columns, 38221 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3713 rows, 6877 columns and 38221 nonzeros
Variable types: 1 continuous, 6876 integer (6876 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3713 rows and 6877 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
```

```
Thread count was 1 (of 8 available processors)

Solution count 1: 824

Optimal solution found (tolerance 2.00e-02)
Best objective 8.240000000000e+02, best bound 8.240000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

Scenario 3 summary As we see below, the addition of TT3 allowed a reduction of 0.5 FTEs and a reduction in the number of understaffed periods. The ability to use different shift lengths will often help in matching highly variable demand patterns.

[21]:
```
ftesum_3 = pd.read_csv('output/scenario3_tt123_phase2_ftesum.csv')
ftesum_3
```

[21]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         64        14720         824     7360.0      46.0    12800

   sched_eff  tot_periods_us         scenario
0   0.869565            33.0  scenario3_tt123
```

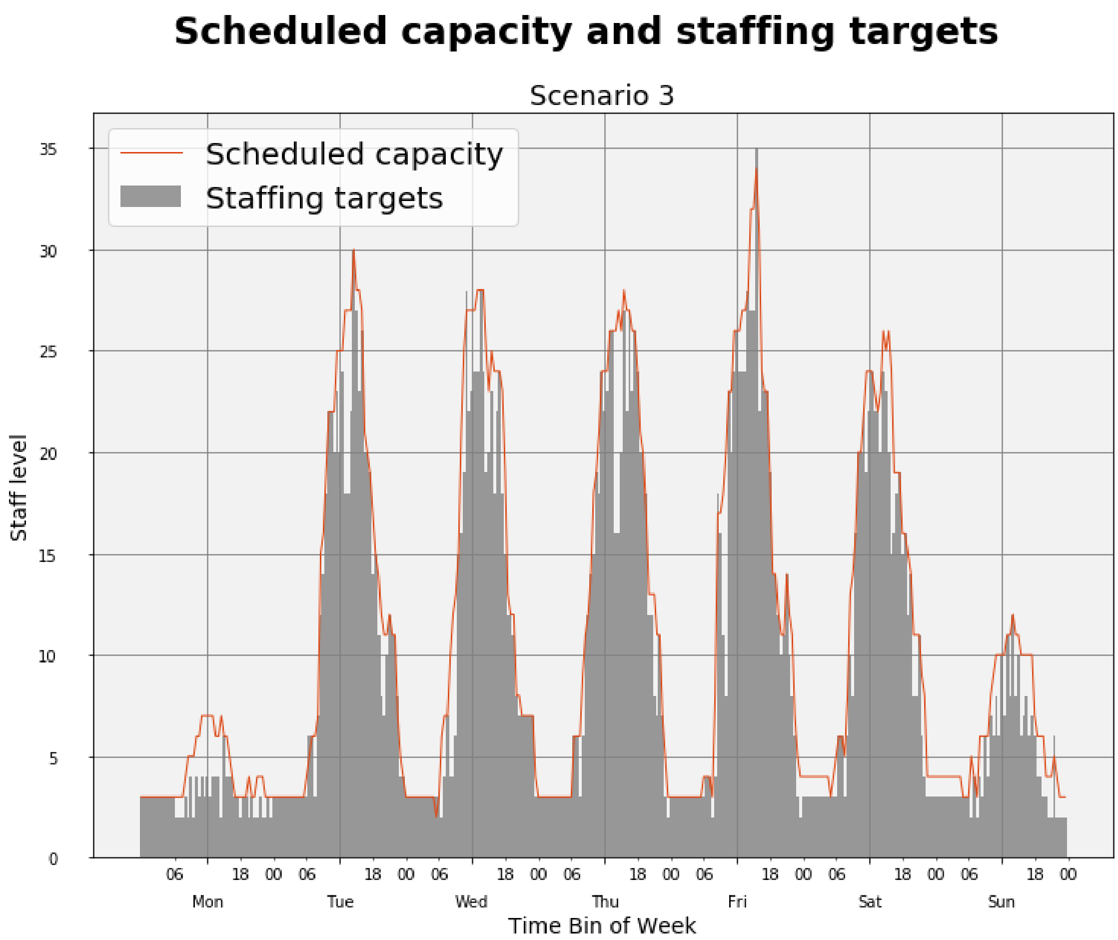
[22]:
```
tourtypesum_3 = pd.read_csv('output/scenario3_tt123_phase2_tourtypesum.csv')
tourtypesum_3
```

[22]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1          4         1280          80      640.0       4.0
1         2         36         5760         360     2880.0      18.0
2         3         24         7680         384     3840.0      24.0

          scenario
0  scenario3_tt123
1  scenario3_tt123
2  scenario3_tt123
```

[23]:
```
# Plot of cap and demand
capacity3_df = pd.read_csv('output/scenario3_tt123_phase1_capsum.csv')
capacity3_df = capacity3_df.loc[capacity3_df['week'] == 1]
capacity3_df = capacity3_df.sort_values(by=['day', 'period'])
```

[24]:
```
# Create a Figure and Axes object and call plot function
fig3 = plt.figure()
```

```
fig3.set_size_inches(12, 9)
fig3.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
↪fontweight='bold')
ax3 = fig3.add_subplot(1,1,1)

capacity_plot(capacity3_df, 'Scenario 3', ax3);
```



1.4.6 Scenario 4 - TT1 and TT3

Finally, let's consider a scenario in which only full-time tours are allowed - TT1 and TT3.

[22]:
```
!pymwts scenario4_tt13 ./input/scenario4_tt13.dat -s gurobi -p ./output/ -t 600␣
↪-g 0.02
```

```
Namespace(mipgap=0.02, path='./output/', phase1dat='./input/scenario4_tt13.dat',
scenario='scenario4_tt13', solver='gurobi', timelimit=600)
```

```
*** Scenario scenario4_tt13


*** Phase 1 model instance created.


*** Setting up the solver.


*** Starting to solve Phase 1 model

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpn_9ebqls.pyomo.lp
Reading time = 0.14 seconds
x22457: 31302 rows, 22457 columns, 412833 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 31302 rows, 22457 columns and 412833 nonzeros
Variable types: 9409 continuous, 13048 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [6e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 22465 rows and 16141 columns
Presolve time: 0.29s
Presolved: 8837 rows, 6316 columns, 57790 nonzeros
Variable types: 1612 continuous, 4704 integer (300 binary)

Root relaxation: objective 1.522959e+04, 10704 iterations, 3.98 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 15229.5871    0 1292          - 15229.5871      -     -    4s
H    0     0                    800548.00000 15229.5871  98.1%     -    5s
     0     0 15229.5871    0 1292 800548.000 15229.5871  98.1%     -    5s
     0     2 15229.5871    0 1292 800548.000 15229.5871  98.1%     -    8s
     7    12 15242.7940    3 1249 800548.000 15233.1321  98.1%   710   10s
    44    47 15258.5027    8 1292 800548.000 15238.0804  98.1%   827   15s
H   60    61                    18434.000000 15238.0804  17.3%   783   16s
    98    98 15271.0482   16 1125 18434.0000 15238.0804  17.3%   662   20s
H  126   125                    17338.000000 15238.0804  12.1%   562   20s
H  193   194                    15466.000000 15238.0804  1.47%   422   23s
```

```
Explored 198 nodes (94378 simplex iterations) in 23.31 seconds
Thread count was 8 (of 8 available processors)

Solution count 4: 15466 17338 18434 800548

Optimal solution found (tolerance 2.00e-02)
Best objective 1.546600000000e+04, best bound 1.523808035675e+04, gap 1.4737%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmp4prxos3i.pyomo.lp
Reading time = 0.01 seconds
x4231: 3287 rows, 4231 columns, 25287 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3287 rows, 4231 columns and 25287 nonzeros
Variable types: 1 continuous, 4230 integer (4230 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3239 rows and 4079 columns
Presolve time: 0.03s
Presolved: 48 rows, 152 columns, 488 nonzeros
Variable types: 0 continuous, 152 integer (152 binary)

Root relaxation: interrupted, 0 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

H    0     0                     828.0000000  828.00000  0.00%     -    0s

Explored 0 nodes (0 simplex iterations) in 0.04 seconds
Thread count was 8 (of 8 available processors)

Solution count 1: 828
```

```
Optimal solution found (tolerance 2.00e-02)
Best objective 8.280000000000e+02, best bound 8.280000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

Scenario 4 summary The addition of TT3 to TT1 leads to a 1.0 FTE savings from Scenario 1 but is a higher cost solution than Scenario 2. Of course, Scenario 3 will have the lowest cost as it has the maximum level of scheduling flexibility considered in this analysis.

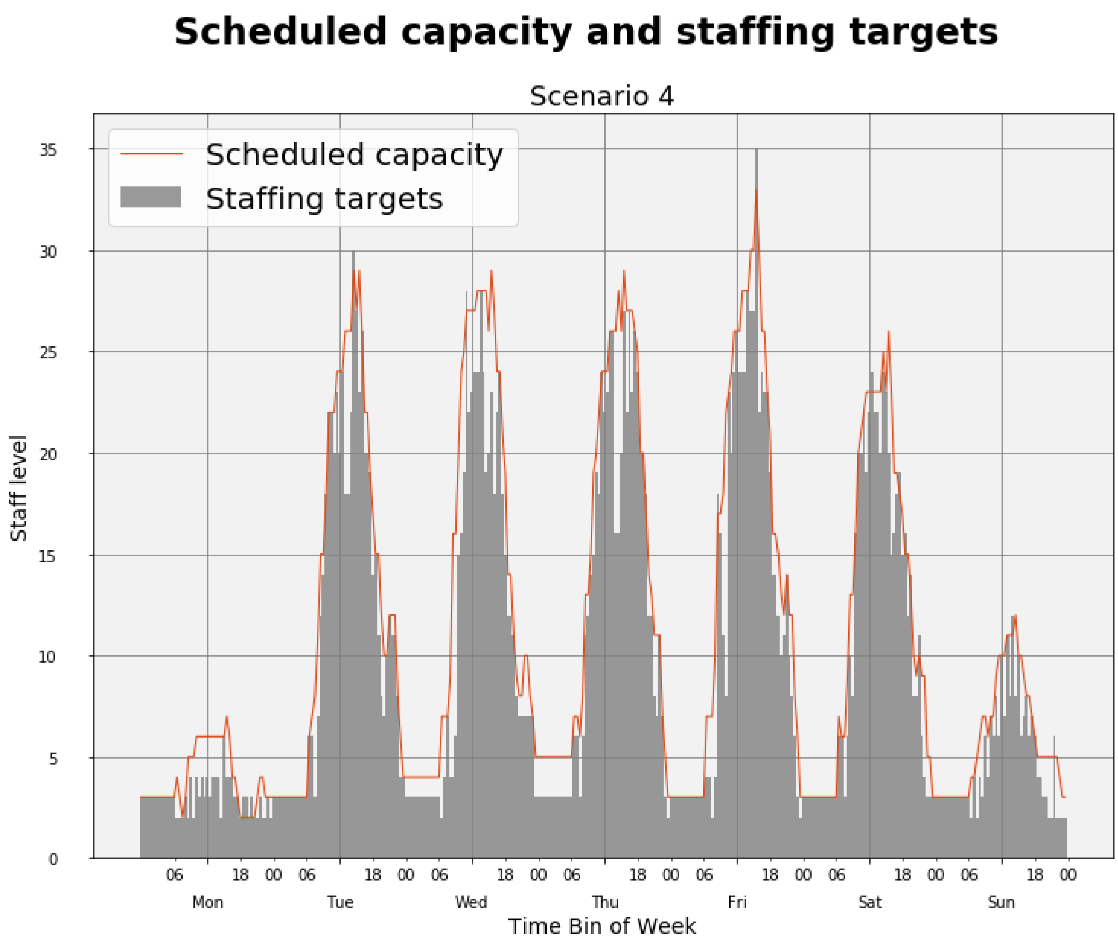
[25]:
```
ftesum_4 = pd.read_csv('output/scenario4_tt13_phase2_ftesum.csv')
```

[26]:
```
tourtypesum_4 = pd.read_csv('output/scenario4_tt13_phase2_tourtypesum.csv')
tourtypesum_4
```

[26]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1         19         6080         380     3040.0      19.0
1         3         28         8960         448     4480.0      28.0

         scenario
0  scenario4_tt13
1  scenario4_tt13
```

[27]:
```
# Plot of cap and demand
capacity4_df = pd.read_csv('output/scenario4_tt13_phase1_capsum.csv')
capacity4_df = capacity4_df.loc[capacity4_df['week'] == 1]
capacity4_df = capacity4_df.sort_values(by=['day', 'period'])
```

[28]:
```
# Create a Figure and Axes object and call plot function
fig4 = plt.figure()
fig4.set_size_inches(12, 9)
fig4.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
 ↪fontweight='bold')
ax4 = fig4.add_subplot(1,1,1)

capacity_plot(capacity4_df, 'Scenario 4', ax4);
```



Let's look at all four FTE summaries:

[29]:
```
ftesum = pd.concat([ftesum_1, ftesum_2, ftesum_3, ftesum_4], ignore_index=True)
ftesum
```

[29]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         48        15360         960     7680.0      48.0    12800
1         65        14880         930     7440.0      46.5    12800
2         64        14720         824     7360.0      46.0    12800
3         47        15040         828     7520.0      47.0    12800

   sched_eff  tot_periods_us         scenario
0   0.833333           168.0     scenario1_tt1
1   0.860215            71.0    scenario2_tt12
2   0.869565            33.0   scenario3_tt123
3   0.851064            67.0    scenario4_tt13
```

No surprises. Scenario 3, which has the highest level of scheduling flexibility, not only has the lowest staffing level (tot_ftes) (equivalently, highest level of scheduling efficiency), it also has the lowest total number of periods of understaffing (tot_periods_us).

1.4.7 Example schedules

Let's look at the actual four-week schedule for Scenario 3.

[30]:
```
schedule3_df = pd.read_csv('output/scenario3_tt123_phase2_mwt.csv')
```

Here's what the raw mwt file looks like. Each row is a tour. There are a few summary values for each tour in the first seven columns. These are followed by the shift worked on each day of the four weeks. An ``x'' signifies not working that day.

[31]:
```
schedule3_df
```

[31]:
```
    tournum  tourtype  tot_shifts  tot_periods  startwin  tot_hours  tot_ftes  \
0         1         2          10          160        11       80.0       0.5
1         2         3          16          320        11      160.0       1.0
2         3         2          10          160        12       80.0       0.5
3         4         3          16          320        12      160.0       1.0
4         5         3          16          320        13      160.0       1.0
..      …         …           …          …        …         …       …
59       60         2          10          160        43       80.0       0.5
60       61         3          16          320        43      160.0       1.0
61       62         2          10          160        44       80.0       0.5
62       63         2          10          160        47       80.0       0.5
63       64         2          10          160        48       80.0       0.5

   Su-1       Mo-1       Tu-1  …       Th-3       Fr-3 Sa-3 Su-4       Mo-4  \
0     x  0500-1300  0500-1300  …          x          x    x    x  0500-1300
1     x          x  0500-1500  …  0500-1500          x    x    x  0500-1500
2     x          x  0530-1330  …          x  0530-1330    x    x  0530-1330
3     x          x  0530-1530  …  0530-1530  0530-1530    x    x  0530-1530
4     x          x  0600-1600  …  0600-1600  0600-1600    x    x  0600-1600
..  …        …        …  …        …        …  …  …        …
59    x          x          x  …          x  2100-0500    x    x  2100-0500
60    x  2100-0700  2100-0700  …  2100-0700          x    x    x  2100-0700
61    x          x  2130-0530  …          x  2130-0530    x    x          x
62    x  2300-0700          x  …          x  2300-0700    x    x  2300-0700
63    x  2330-0730          x  …  2330-0730          x    x    x  2330-0730

         Tu-4       We-4       Th-4       Fr-4 Sa-4
0           x          x  0500-1300  0500-1300    x
1   0500-1500  0500-1500          x  0500-1500    x
2           x          x  0530-1330  0530-1330    x
3   0530-1530  0530-1530          x  0530-1530    x
```

```
4   0600-1600  0600-1600  0600-1600          x    x
..         …          …          …      …    …
59          x  2100-0500  2100-0500          x    x
60  2100-0700  2100-0700  2100-0700          x    x
61  2130-0530  2130-0530          x  2130-0530    x
62  2300-0700          x  2300-0700          x    x
63  2330-0730  2330-0730          x          x    x

[64 rows x 35 columns]
```

Now we'll add a little styling to the schedule.

[32]:
```
# Use tournum for the index
schedule3_df.set_index('tournum', inplace=True)
# Create a list of column indices to display. Just display tour type and the␣
↪schedule.
col_list = [0]
col_list = col_list + [i for i in range(6, 34)]
```

Several different ways to do styling globally and based on cell value (conditional formatting). The following links are useful.

https://pandas.pydata.org/pandas-docs/stable/user_guide/style.html
https://python-graph-gallery.com/python-colors/

[33]:
```
def highlight(s):
    """
    Highlight tour row based on tour type.
    """
    if s.tourtype == 1:
        return ['background-color: wheat']*len(s)
    elif s.tourtype == 2:
        return ['background-color: khaki']*len(s)
    else:
        return ['background-color: beige']*len(s)
```

[34]:
```
# Set table level styles by first creating list of dictionaries with CSS␣
↪selector as key and
# list of property tuples
table_styles = [
    dict(selector="th", props=[("font-size", "6pt"),
                               ("text-align", "center")]),
    dict(selector="td", props=[("font-size", "6pt"),
                               ("text-align", "center")])
]

# Apply the table styles and row highlighting
```

```
schedule3_df.iloc[:, col_list].style.set_table_styles(table_styles).
 ↪apply(highlight, axis=1)
```

[34]: <pandas.io.formats.style.Styler at 0x7fb2baf80490>

[35]:
```
schedule3_html = schedule3_df.iloc[:, col_list].style.
 ↪set_table_styles(table_styles).apply(highlight, axis=1).render()
```

[36]:
```
with open('schedule3.html', "w") as f:
     f.write(schedule3_html)
```

1.5 Scenario 3a - Enhancements to Scenario 3

Let's allow just a little intra-tour start time flexibility. In particular, shift start times can vary by one half-hour from day to day. For example, a tour assigned to the start window at 8a, could have shifts that start at 8a or 8:30a every day.

[37]:
```
!pymwts scenario3a_tt123 ./input/scenario3a_tt123.dat -s gurobi -p ./output/ -t␣
 ↪600 -g 0.02
```

```
Namespace(mipgap=0.02, path='./output/',
phase1dat='./input/scenario3a_tt123.dat', scenario='scenario3a_tt123',
solver='gurobi', timelimit=600)

*** Scenario scenario3a_tt123


*** Phase 1 model instance created.


*** Setting up the solver.


*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpw75hp8xq.pyomo.lp
Reading time = 0.26 seconds
x22321: 30657 rows, 22321 columns, 557807 nonzeros
Changed value of parameter mipgap to 0.02
```

```
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 30657 rows, 22321 columns and 557807 nonzeros
Variable types: 9409 continuous, 12912 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [6e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 14485 rows and 11170 columns
Presolve time: 1.11s
Presolved: 16172 rows, 11151 columns, 300420 nonzeros
Variable types: 1420 continuous, 9731 integer (396 binary)

Deterministic concurrent LP optimizer: primal and dual simplex
Showing first log only…


Root simplex log…

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   10570    1.2308243e+05   0.000000e+00   5.635881e+06      5s
   18334    8.1152702e+04   0.000000e+00   4.741168e+07     10s
   25014    3.7454331e+04   0.000000e+00   1.111644e+08     15s
   30472    1.9434494e+04   0.000000e+00   1.743091e+07     20s
   35065    1.5455226e+04   0.000000e+00   2.217404e+06     25s
   38665    1.4886805e+04   0.000000e+00   1.495464e+06     30s
   41723    1.4599019e+04   0.000000e+00   4.921192e+05     35s
   44814    1.4447573e+04   0.000000e+00   6.775631e+05     40s
   47951    1.4385644e+04   0.000000e+00   3.494779e+05     45s
   50360    1.4365768e+04   0.000000e+00   7.635857e+05     50s
   53427    1.4353376e+04   0.000000e+00   1.751655e+06     55s
Concurrent spin time: 0.01s

Solved with dual simplex

Root relaxation: objective 1.433427e+04, 63680 iterations, 53.63 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 14334.2661    0 1911          - 14334.2661      -     -   56s
H    0     0                      130848.00000 14334.2661  89.0%     -   61s
     0     0 14334.2661    0 1911 130848.000 14334.2661  89.0%     -   62s
     0     2 14334.2661    0 1911 130848.000 14334.2661  89.0%     -   73s
     1     4 14334.3259    1 1811 130848.000 14334.2752  89.0%  1430   75s
     7    12 14334.8670    3 1783 130848.000 14334.8670  89.0%  2024   80s
```

```
    11    16 14334.9202    4 1816 130848.000 14334.8670  89.0%  1533   86s
    19    21 14335.1223    5 1873 130848.000 14334.8670  89.0%  2194   94s
    23    26 14335.2894    5 1721 130848.000 14334.8670  89.0%  2364   97s
    28    26 14338.4167    6 1757 130848.000 14334.8670  89.0%  2346  103s
    32    32 14335.9225    6 1748 130848.000 14334.8670  89.0%  2254  105s
    42    43 14341.1095    7 1809 130848.000 14334.8670  89.0%  2054  114s
    47    47 14341.1095    8 1788 130848.000 14334.8670  89.0%  1920  120s
    61    63 14346.2632   11 1633 130848.000 14334.8670  89.0%  1865  127s
    65    64 14346.5702   12 1706 130848.000 14334.8670  89.0%  1865  130s
    83    87 14347.9318   15 1479 130848.000 14334.8670  89.0%  1618  137s
    95    97 14349.8634   18 1511 130848.000 14334.8670  89.0%  1540  141s
   104   105 14353.5090   20 1725 130848.000 14334.8670  89.0%  1589  147s
   108   110 14351.5625   20 1619 130848.000 14334.8670  89.0%  1638  150s
   125   126 14352.0909   24 1460 130848.000 14334.8670  89.0%  1554  157s
   132   133 14352.3137   26 1464 130848.000 14334.8670  89.0%  1586  165s
   142   143 14352.3997   28 1454 130848.000 14334.8670  89.0%  1726  170s
   167   168 14354.5903   34 1558 130848.000 14334.8670  89.0%  1630  177s
   181   182 14354.5903   36 1558 130848.000 14334.8670  89.0%  1572  181s
   189   192 14354.5903   37 1558 130848.000 14334.8670  89.0%  1575  189s
H  190   192                    14940.000000 14334.8670  4.05%  1567  189s
   200   201 14355.1487   39 1499 14940.0000 14334.8670  4.05%  1582  195s
   224   226 14355.4874   41 1609 14940.0000 14334.8670  4.05%  1523  210s
H  230   229                    14504.000000 14334.8670  1.17%  1508  210s

Explored 234 nodes (419047 simplex iterations) in 210.37 seconds
Thread count was 8 (of 8 available processors)

Solution count 3: 14504 14940 130848

Optimal solution found (tolerance 2.00e-02)
Best objective 1.450400000000e+04, best bound 1.433486704482e+04, gap 1.1661%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpchw2iyk4.pyomo.lp
Reading time = 0.05 seconds
```

```
x8402: 3597 rows, 8402 columns, 53166 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3597 rows, 8402 columns and 53166 nonzeros
Variable types: 1 continuous, 8401 integer (8401 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3597 rows and 8402 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 8 available processors)

Solution count 1: 810

Optimal solution found (tolerance 2.00e-02)
Best objective 8.100000000000e+02, best bound 8.100000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

[38]:
```python
ftesum_3a = pd.read_csv('output/scenario3a_tt123_phase2_ftesum.csv')
```

[40]:
```python
ftesum_3a
```

[40]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         62        14240         810     7120.0      44.5    12800

   sched_eff  tot_periods_us          scenario
0   0.898876            42.0  scenario3a_tt123
```

[39]:
```python
tourtypesum_3a = pd.read_csv('output/scenario3a_tt123_phase2_tourtypesum.csv')
tourtypesum_3a
```

[39]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1          7         2240         140     1120.0       7.0
1         2         35         5600         350     2800.0      17.5
2         3         20         6400         320     3200.0      20.0
```

```
            scenario
0  scenario3a_tt123
1  scenario3a_tt123
2  scenario3a_tt123
```

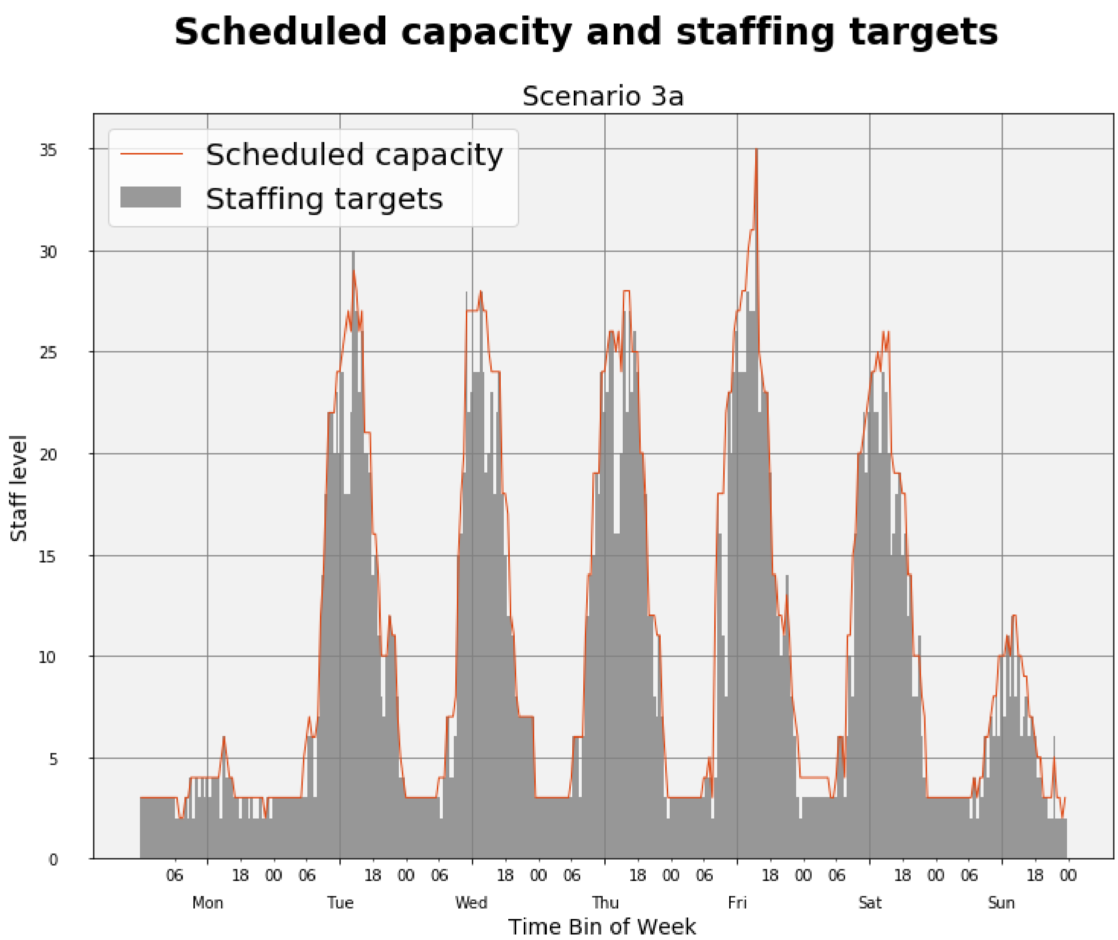
Adding just a little bit of intra-tour start time flexibility led to a savings of 1.5 FTEs at the cost of a small amount of additinal understaffing (an increase of nine half-hours over four weeks).

[41]:
```
ftesum = pd.concat([ftesum_1, ftesum_2, ftesum_3, ftesum_4, ftesum_3a],␣
 ↪ignore_index=True)
ftesum
```

[41]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         48        15360         960     7680.0      48.0    12800
1         65        14880         930     7440.0      46.5    12800
2         64        14720         824     7360.0      46.0    12800
3         47        15040         828     7520.0      47.0    12800
4         62        14240         810     7120.0      44.5    12800

   sched_eff  tot_periods_us          scenario
0   0.833333           168.0    scenario1_tt1
1   0.860215            71.0   scenario2_tt12
2   0.869565            33.0  scenario3_tt123
3   0.851064            67.0   scenario4_tt13
4   0.898876            42.0 scenario3a_tt123
```

[44]:
```
# Plot of cap and demand
capacity3a_df = pd.read_csv('output/scenario3a_tt123_phase1_capsum.csv')
capacity3a_df = capacity3a_df.loc[capacity3a_df['week'] == 1]
capacity3a_df = capacity3a_df.sort_values(by=['day', 'period'])
```

[46]:
```
# Create a Figure and Axes object and call plot function
fig3a = plt.figure()
fig3a.set_size_inches(12, 9)
fig3a.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
 ↪fontweight='bold')
ax3a = fig3a.add_subplot(1,1,1)

capacity_plot(capacity3a_df, 'Scenario 3a', ax3a);
```



Now let's see what happens if still allow the same amount of intra-tour start time flexbility as Scenario 3a, but we set the understaffing costs extremely high to eliminate all understaffing and treating the staffing level targets as hard lower bounds. We'll call this Scenario 3b.

[47]:
```
!pymwts scenario3b_tt123 ./input/scenario3b_tt123.dat -s gurobi -p ./output/ -t␣
↪600 -g 0.02
```

```
Namespace(mipgap=0.02, path='./output/',
phase1dat='./input/scenario3b_tt123.dat', scenario='scenario3b_tt123',
solver='gurobi', timelimit=600)

*** Scenario scenario3b_tt123


*** Phase 1 model instance created.


*** Setting up the solver.
```

```
*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpno730izn.pyomo.lp
Reading time = 0.19 seconds
x22321: 30657 rows, 22321 columns, 557807 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 30657 rows, 22321 columns and 557807 nonzeros
Variable types: 9409 continuous, 12912 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [2e+01, 5e+04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 14485 rows and 11170 columns
Presolve time: 1.19s
Presolved: 16172 rows, 11151 columns, 300420 nonzeros
Variable types: 1420 continuous, 9731 integer (396 binary)

Deterministic concurrent LP optimizer: primal and dual simplex
Showing first log only…


Root simplex log…

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   11481   -2.9808582e+09   2.272216e+02   4.026320e+10      5s
   17166    5.1464160e+08   0.000000e+00   3.231368e+11      8s
   19886    4.8903295e+08   0.000000e+00   3.997527e+10     10s
   26564    1.3432775e+08   0.000000e+00   8.014929e+10     15s
   34360    9.5062582e+04   0.000000e+00   6.147065e+06     20s
   39098    9.2768050e+04   0.000000e+00   4.627427e+06     25s
   43088    9.2105812e+04   0.000000e+00   3.647282e+06     30s
   46874    9.1472374e+04   0.000000e+00   2.136969e+06     35s
Concurrent spin time: 0.00s

Solved with dual simplex
```

```
Root relaxation: objective 1.449067e+04, 36775 iterations, 37.39 seconds
Total elapsed time = 47.71s
Total elapsed time = 51.25s
Total elapsed time = 56.87s

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 14490.6667    0 1130          - 14490.6667      -     -   58s
H    0     0                    5.182419e+08 14490.6667   100%     -   61s
     0     0 14490.6667    0 1130 5.1824e+08 14490.6667   100%     -   64s
     0     2 14490.6667    0 1130 5.1824e+08 14490.6667   100%     -   96s
     1     4 14490.6667    1 1366 5.1824e+08 14490.6667   100%  6197  102s
     3     8 14490.6667    2 1268 5.1824e+08 14490.6667   100%  4419  110s
     7    12 14490.6667    3 1784 5.1824e+08 14490.6667   100%  4897  118s
    11    16 14490.6667    3 1823 5.1824e+08 14490.6667   100%  5873  143s
    15    20 14490.6667    4 1538 5.1824e+08 14490.6667   100%  8533  159s
    19    18 14490.6667    5 1447 5.1824e+08 14490.6667   100%  8327  162s
    23    23 14490.6667    5 1560 5.1824e+08 14490.6667   100%  7189  165s
    28    29 14490.6667    6 1592 5.1824e+08 14490.6667   100%  6214  176s
    33    34 14490.6667    7 1473 5.1824e+08 14490.6667   100%  6027  180s
    39    40 14490.6667    8 1167 5.1824e+08 14490.6667   100%  5227  185s
    49    52 14490.6667    9 1479 5.1824e+08 14490.6667   100%  4536  195s
    58    61 14490.6667    9 1680 5.1824e+08 14490.6667   100%  4322  210s
H   59    61                    204560.00000 14490.6667  92.9%  4249  210s
    69    71 14492.9524   10 1525 204560.000 14490.6667  92.9%  4113  225s
    73    68 14492.9524   11 1386 204560.000 14490.6667  92.9%  4161  238s
    89    94 14493.0909   14 1232 204560.000 14490.6667  92.9%  3866  254s
H   97    96                    64560.000000 14490.6667  77.6%  3644  254s
   111   111 14493.0909   16 1084 64560.0000 14490.6667  77.6%  3452  268s
   132   133 14493.7143   19 1046 64560.0000 14490.6667  77.6%  3208  282s
H  139   139                    14560.000000 14490.6667  0.48%  3124  282s

Explored 150 nodes (527364 simplex iterations) in 282.24 seconds
Thread count was 8 (of 8 available processors)

Solution count 4: 14560 64560 204560 5.18242e+08

Optimal solution found (tolerance 2.00e-02)
Best objective 1.456000000000e+04, best bound 1.449066666667e+04, gap 0.4762%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.
```

```
--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpt077s8u5.pyomo.lp
Reading time = 0.04 seconds
x8520: 3655 rows, 8520 columns, 53936 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3655 rows, 8520 columns and 53936 nonzeros
Variable types: 1 continuous, 8519 integer (8519 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3655 rows and 8520 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 8 available processors)

Solution count 1: 822

Optimal solution found (tolerance 2.00e-02)
Best objective 8.220000000000e+02, best bound 8.220000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

[48]:
```
ftesum_3b = pd.read_csv('output/scenario3b_tt123_phase2_ftesum.csv')
```

[49]:
```
ftesum_3b
```

[49]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         63        14560         822     7280.0      45.5    12800

   sched_eff  tot_periods_us          scenario
```

```
0   0.879121             0.0  scenario3b_tt123
```

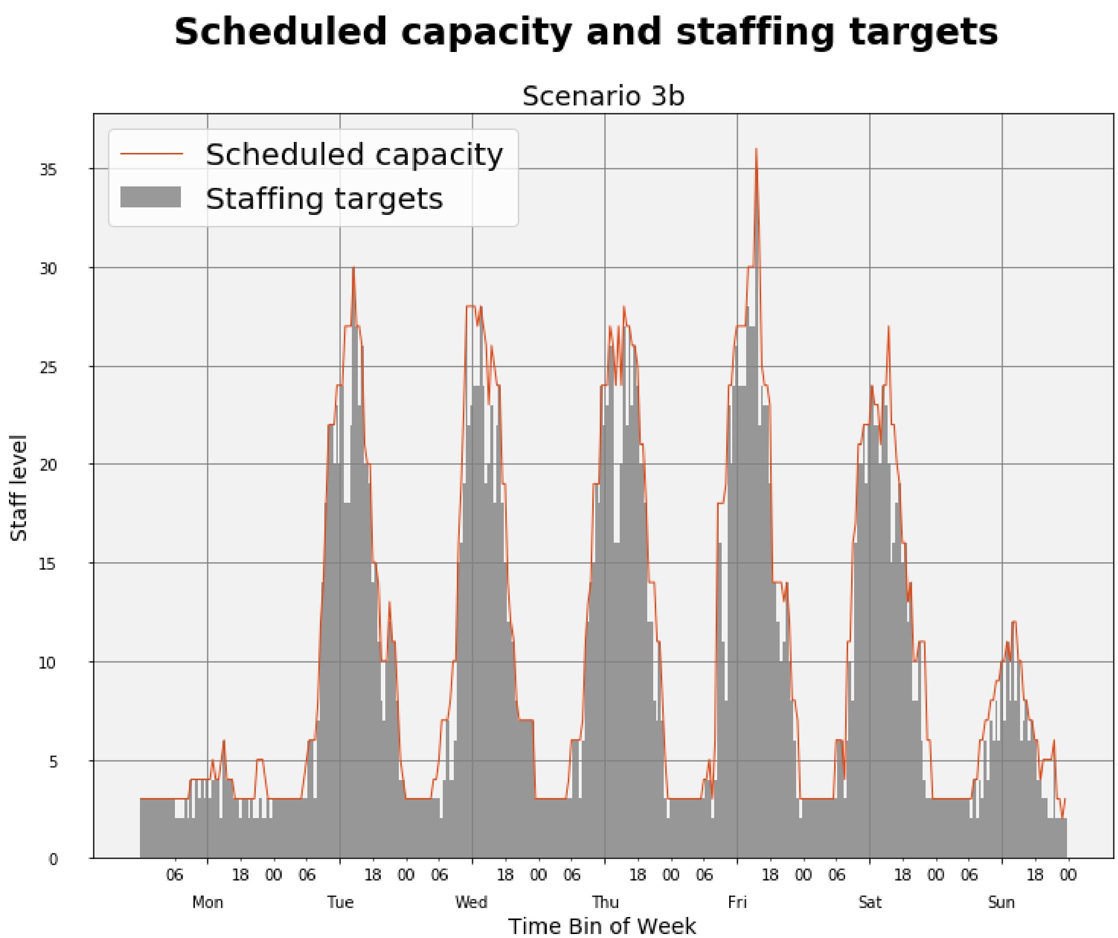
[50]:
```
tourtypesum_3b= pd.read_csv('output/scenario3b_tt123_phase2_tourtypesum.csv')
tourtypesum_3b
```

[50]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1          6         1920         120      960.0       6.0
1         2         35         5600         350     2800.0      17.5
2         3         22         7040         352     3520.0      22.0

           scenario
0  scenario3b_tt123
1  scenario3b_tt123
2  scenario3b_tt123
```

[52]:
```
ftesum = pd.concat([ftesum_3, ftesum_3a, ftesum_3b], ignore_index=True)
ftesum
```

[52]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         64        14720         824     7360.0      46.0    12800
1         62        14240         810     7120.0      44.5    12800
2         63        14560         822     7280.0      45.5    12800

   sched_eff  tot_periods_us          scenario
0   0.869565            33.0   scenario3_tt123
1   0.898876            42.0  scenario3a_tt123
2   0.879121             0.0  scenario3b_tt123
```

[53]:
```
# Plot of cap and demand
capacity3b_df = pd.read_csv('output/scenario3b_tt123_phase1_capsum.csv')
capacity3b_df = capacity3b_df.loc[capacity3b_df['week'] == 1]
capacity3b_df = capacity3b_df.sort_values(by=['day', 'period'])

# Create a Figure and Axes object and call plot function
fig3b = plt.figure()
fig3b.set_size_inches(12, 9)
fig3b.suptitle('Scheduled capacity and staffing targets', fontsize=24,␣
 ↪fontweight='bold')
ax3b = fig3b.add_subplot(1,1,1)

capacity_plot(capacity3b_df, 'Scenario 3b', ax3b);
```



So, allowing no understaffing led to an increase of 1.0 FTEs from Scenario 3a.

Let's run one more scenario with no understaffing but an increase in intra-tour start time flexibility to four periods. So, a tour assigned to a start time window of 8a could have shift start times of 8a, 8:30a, 9a, or 9:30a.

[54]:
```
!pymwts scenario3c_tt123 ./input/scenario3c_tt123.dat -s gurobi -p ./output/ -t␣
↪600 -g 0.02
```

```
Namespace(mipgap=0.02, path='./output/',
phase1dat='./input/scenario3c_tt123.dat', scenario='scenario3c_tt123',
solver='gurobi', timelimit=600)

*** Scenario scenario3c_tt123


*** Phase 1 model instance created.
```

```
*** Setting up the solver.


*** Starting to solve Phase 1 model


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpmxdvlfgk.pyomo.lp
Reading time = 0.20 seconds
x20769: 28257 rows, 20769 columns, 501655 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 28257 rows, 20769 columns and 501655 nonzeros
Variable types: 9409 continuous, 11360 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [2e+01, 5e+04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+05]
Presolve removed 14092 rows and 10470 columns
Presolve time: 1.19s
Presolved: 14165 rows, 10299 columns, 262687 nonzeros
Variable types: 1420 continuous, 8879 integer (396 binary)

Deterministic concurrent LP optimizer: primal and dual simplex
Showing first log only…


Root simplex log…

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   11856    3.5774130e+08   0.000000e+00   5.850829e+11      5s
   20608    4.6288931e+05   0.000000e+00   1.374674e+09     10s
   27673    8.0294479e+04   0.000000e+00   2.140780e+06     15s
   32333    7.9198943e+04   0.000000e+00   1.746470e+06     20s
   36177    7.8903223e+04   0.000000e+00   3.988148e+05     25s
Concurrent spin time: 0.00s

Solved with dual simplex

Root relaxation: objective 1.424446e+04, 28393 iterations, 27.35 seconds
Total elapsed time = 31.49s
```

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 14244.4620    0 1045          - 14244.4620      -     -   34s
H    0     0                    5.146422e+08 14244.4620   100%     -   36s
     0     0 14244.4620    0 1030 5.1464e+08 14244.4620   100%     -   38s
H    0     0                    24880.000000 14244.4620  42.7%     -   42s
H    0     0                    14560.000000 14244.4620  2.17%     -   44s
     0     0 14244.4620    0  916 14560.0000 14244.4620  2.17%     -   64s
     0     0 14244.4620    0  956 14560.0000 14244.4620  2.17%     -   67s
     0     0 14244.4620    0  990 14560.0000 14244.4620  2.17%     -   69s
     0     0 14245.0042    0 1150 14560.0000 14245.0042  2.16%     -   74s
     0     0 14245.0042    0 1247 14560.0000 14245.0042  2.16%     -   76s
     0     0 14245.0042    0 1247 14560.0000 14245.0042  2.16%     -   76s
     0     0 14245.0042    0 1214 14560.0000 14245.0042  2.16%     -   80s
     0     0 14245.0042    0 1239 14560.0000 14245.0042  2.16%     -   81s
     0     0 14245.0042    0 1225 14560.0000 14245.0042  2.16%     -   87s
     0     0 14245.0042    0 1222 14560.0000 14245.0042  2.16%     -   88s
     0     0 14245.8403    0 1097 14560.0000 14245.8403  2.16%     -   94s
H    0     0                    14400.000000 14245.8403  1.07%     -   95s

Cutting planes:
  Zero half: 4

Explored 1 nodes (97398 simplex iterations) in 95.85 seconds
Thread count was 8 (of 8 available processors)

Solution count 4: 14400 14560 24880 5.14642e+08

Optimal solution found (tolerance 2.00e-02)
Best objective 1.440000000000e+04, best bound 1.424584030418e+04, gap 1.0706%

*** Phase 1 solution found


*** Phase 2 model instance created.


*** Starting to solve Phase 2 model.


--------------------------------------------
Warning: your license will expire in 1 days
--------------------------------------------

Academic license - for non-commercial use only
Read LP format model from file /tmp/tmpwz5si0jx.pyomo.lp
```

```
Reading time = 0.04 seconds
x11607: 3539 rows, 11607 columns, 83651 nonzeros
Changed value of parameter mipgap to 0.02
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter timelimit to 600.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 3539 rows, 11607 columns and 83651 nonzeros
Variable types: 1 continuous, 11606 integer (11606 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 3539 rows and 11607 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.03 seconds
Thread count was 1 (of 8 available processors)

Solution count 1: 812

Optimal solution found (tolerance 2.00e-02)
Best objective 8.120000000000e+02, best bound 8.120000000000e+02, gap 0.0000%

*** Phase 2 model solved.


*** Output files created.
```

[55]:
```
ftesum_3c = pd.read_csv('output/scenario3c_tt123_phase2_ftesum.csv')
```

[57]:
```
ftesum_3c
```

[57]:
```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         61        14400         812     7200.0      45.0    12800

   sched_eff  tot_periods_us          scenario
0   0.888889             0.0  scenario3c_tt123
```

[58]:
```
tourtypesum_3c = pd.read_csv('output/scenario3c_tt123_phase2_tourtypesum.csv')
tourtypesum_3c
```

[58]:
```
   tourtype  num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  \
0         1          7         2240         140     1120.0       7.0
1         2         32         5120         320     2560.0      16.0
```

```
2         3          22          7040          352      3520.0       22.0

           scenario
0  scenario3c_tt123
1  scenario3c_tt123
2  scenario3c_tt123
```

[60]:

```python
ftesum = pd.concat([ftesum_1, ftesum_2, ftesum_3, ftesum_3a, ftesum_3b,
 ↪ftesum_3c], ignore_index=True)
ftesum
```

[60]:

```
   num_tours  tot_periods  tot_shifts  tot_hours  tot_ftes  tot_dmd  \
0         48        15360         960     7680.0      48.0    12800
1         65        14880         930     7440.0      46.5    12800
2         64        14720         824     7360.0      46.0    12800
3         62        14240         810     7120.0      44.5    12800
4         63        14560         822     7280.0      45.5    12800
5         61        14400         812     7200.0      45.0    12800

   sched_eff  tot_periods_us          scenario
0   0.833333           168.0     scenario1_tt1
1   0.860215            71.0    scenario2_tt12
2   0.869565            33.0   scenario3_tt123
3   0.898876            42.0  scenario3a_tt123
4   0.879121             0.0  scenario3b_tt123
5   0.888889             0.0  scenario3c_tt123
```

1.6 The bottom line

A more informed choice could now be made about the tradeoffs between scheduling flexibility, the amount of understaffing, and total labor costs. This is what tactical scheduling analysis is all about and for which the pymwts model was developed.

[ ]:

```python

```