IEEE-CIS TECHNICAL CHALLENGE ON PREDICT+OPTIMIZE FOR RENEWABLE ENERGY SCHEDULING

DATA DESCRIPTION AND SUBMISSION

V1.0, July 02, 2021

https://ieee-dataport.org/competitions/ieee-cis-technical-challenge-predictoptimize-renewable-energy-scheduling

1 Data Description

The goal of this competition is to develop an optimal battery schedule and an optimal lecture schedule. For that, you are expected to forecast the energy demand and power production of a set of buildings and solar panels beforehand.

You are provided with historical 15-minutely energy demand of 6 buildings and 15-minutely power production of 6 solar panels from Monash University, Melbourne, Australia. The starting timestamps of each of these 12 time series are different from each other. However, each time series contains recordings until 30th September 2020. In the dataset, nil values are registered for the time slots where the energy demand or solar power production are not recorded.

Furthermore, we encourage you to use the following data sources, which we regretfully cannot provide as part of the competition dataset.

Weather data: we provide an R script (*write_tsf_weather_data.R*) to download the daily minimum temperature (C), maximum temperature (C), rainfall (mm) and solar exposure (MJm^{-2}) of three weather stations near Melbourne: Olympic Park, Moorabbin Airport and Oakleigh (Metropolitan Golf Club) from http://www.bom.gov.au/climate/data/. Each weather series starts from 1st of January 2016 and contains corresponding values probably up to the date you download the data.

Electricity price data: as the final goal is scheduling to achieve the lowest energy cost, you should use the Australian electricity price data available at: https://www.aemo.com.au/energy-systems/electricity/national-electricity-market-nem/data-nem/aggregated-data. In particular, for Phase 1 of the competition, you will need the data for October 2020 for Victoria, that you can download from here: https://www.aemo.com.au/aemo/data/nem/priceanddemand/PRICE_AND_DEMAND_202010_VIC1.csv. This data has time stamps in the AEST zone, Australian Eastern Standard Time.

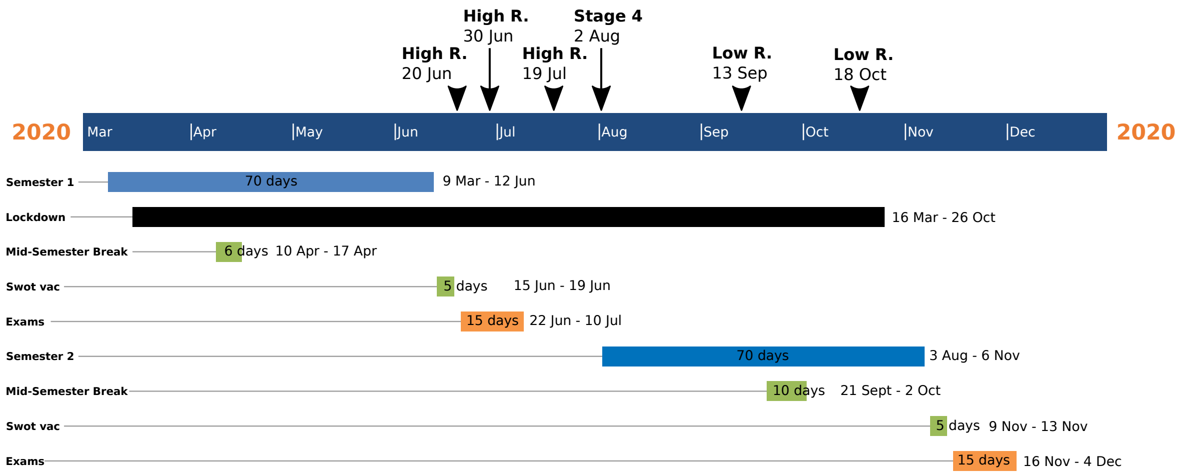
2 Forecasting Problem

For Phase 1 of this competition, you are expected to predict the 15-minutely energy demand of the given 6 buildings and the 15-minutely power production of the given 6 solar panels for all 31 days of October 2020. Therefore, you need to provide 2976 15-minutely period forecasts for all 12 time series given.

At the end of Phase 1, we will release the actual energy demand/power production of all 12 series corresponding with October 2020. Then, for Phase 2, you are expected to provide the 15-minutely energy demand of the given 6 buildings and the 15-minutely power production of the given 6 solar panels for all 30 days of November 2020. Therefore, you need to provide 2880 15-minutely periods ahead forecasts for all 12 time series given.

The final evaluation of the competition and determination of prizes will be based on Phase 2 alone.

We assume that the load that we predict is the base load of the campus. As Melbourne was in lockdown due to the pandemic for large parts of 2020, and the campus was nearly empty, this is a reasonable assumption. A timeline of events with an estimation from us about at which capacity the campus was operating at each time can be found in Figure 1.



Restrictions

Date	Details	Monash Occupancy (%)
20th June	Re-tightened restrictions on household gatherings	10
30th June	Residents need to comply with the 4 acceptable reasons to leave their houses: shopping for essentials, medical/compassionate needs, exercising (only 2 people at once) and work/education purposes.	10
19th July	Mandatory to wear a face mask when going out.	10
2nd August	Stage 4 restrictions: in addition to the above restrictions, curfew was imposed across Melbourne from 8 pm to 5 am and 5km radius restriction was applied for travelling needs.	5
13th September	Reduction of the curfew, some loosening of rules around outdoor exercise and social interactions. Also, students can attend universities for onsite learning.	25
18th October	Reopening of most businesses to the public, increased seating for hospitality, the allowance of visitors for all residents and the resumption of some indoor sports.	30

Figure 1: Timeline of Melbourne lockdown measures in 2020 due to the pandemic

3 Time Series Format

The datasets are provided in the *.tsf* file format, introduced by Godahewa et al. Figure 2 illustrates the Phase 1 training dataset stored in this format in file *phase_1_data.tsf*. Note that the data has UTC time stamps, while Melbourne is in the AEDT time zone (Australian Eastern Daylight Time).

This file first displays a set of tags describing attributes and the meta-information of the dataset such as @frequency (seasonality), @missing (whether the series contain missing values) and @equallength (whether the series have equal lengths). For this dataset threre are 2 attributes: series_name (name of the series) and start_timestamp (starting timestamp of the series).

```
# Dataset Information
# Phase 1 training data of IEEE CIS Technical Challenge 2021
#
@relation energy_demand
@attribute series_name string
@attribute start_timestamp date
@frequency 15_minutes
@missing true
@equallength false
@data
Building0:2016-07-03 21-30-00:283.8,283.8,283.8,606,606,606,606,306,306,306,306,317.8,317.
Building1:2019-01-09 23-15-00:8.1,15.7,22.8,32.7,8.1,16.5,24.7,34.5,8.2,16,24.5,32.5,7.2,1
Building3:2016-03-01 04-15-00:1321,1321,1321,1321,1293,1293,1293,1293,1249,1249,1249,1249,
Building4:2019-07-03 04-45-00:2,?,1,2,?,2,?,?,2,1,1,1,1,1,1,1,1,1,2,2,?,2,1,1,1,2,?,2,2,1,
Building5:2019-07-25 23-00-00:30,31,24,34,30,31,26,33,28,33,27,31,28,33,26,34,27,35,28,25,
Building6:2019-07-25 01-45-00:36.8,34.6,34.6,36.2,36.2,35.2,35.2,35.8,35.8,36,36,36,36,36,
Solar0:2020-04-25 14-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Solar1:2018-12-31 13-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.07,0.14,0.28,0.
Solar2:2019-06-05 14-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Solar3:2019-06-05 14-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Solar4:2019-06-05 14-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Solar5:2019-01-15 13-00-00:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

Figure 2: Phase 1 training dataset stored in phase_1_data.tsf

The data are then listed under the @data tag where one line contains the information of one time series. The 12 lines under the @data tag are corresponding with the 12 time series where 6 of them represent the energy demand of 6 buildings and the other 6 series show the power production of 6 solar panels. Each time series first defines its attribute values that are separated by colons which are the name of the series and starting timestamp. The values of time series are then appended to their attribute vector as a comma-separated variable-length vector. The missing values in the series are indicated using the "?" symbol.

4 R and Python Scripts

We provide the following R and Python scripts for working with the competition data.

4.1 Loading Data

You can use one of the scripts, **data_loader.R** or **data_loader.py**, available at https://github.com/rakshitha123/TSForecasting/tree/master/utils, to load the data stored in the .tsf format into R or Python. The R script converts the data into a tsibble or a dataframe whereas the Python script converts the data into a pandas dataframe.

Please refer to the examples of loading data shown in the scripts for more information.

4.2 Downloading Weather Data

We provide a script **write_tsf_weather_data.R** that downloads the weather data from http://www.bom.gov.au/climate/data/ and converts the downloaded data into a single file called *weather_data.tsf*. The created *weather_data.tsf* file will be stored in the R working directory and should look like shown in Figure 3.

This file also contains the above explained .tsf format. Under the @data tag, there are information on 10 time series: 2 each for maximum and minimum temperatures and 3 each for rainfall and solar exposure. Each time series has 4 attributes: series_name (name of the series), start_timestamp (starting timestamp of the series), type (type of series such as max_temp, min_temp, rainfall or solar) and area (weather station corresponding with the series which is one of olympic, moorabbin and oakleigh). The values of each series are recorded as a comma-separated variable length vector after showing the colon separated 4 attribute values corresponding with them.

```
# Dataset Information
# This file contains information on 4 weather variables: maximum temperature, minimum temperature, rainfall
# and solar exposure corresponding with 3 areas: Melbourne (Olympic Park), Oakleigh (Metropolitan Golf Club)
# and Moorabbin Airport.
#
@relation weather
@attribute series_name string
@attribute start_timestamp date
@attribute type string
@attribute area string
@frequency daily
@missing true
@data
T1:2016-01-01 00-00-00:max_temp:olympic:24.2,27.6,27.1,26,25.7,25.7,21.3,20.6,21.4,31.5,36.4,24.8,42.2,18.1,
T2:2016-01-01 00-00-00:max_temp:moorabbin:25.7,28.5,25.7,25.6,26.9,26.3,22.4,21.2,22.3,32.4,38.2,27.8,42.9,1
T3:2016-01-01 00-00-00:min_temp:olympic:19.8,17.1,17.5,16.3,17.1,17.3,16.3,16.2,16.8,15.7,17.2,17.1,16.8,16.
T4:2016-01-01 00-00-00:min_temp:moorabbin:19.3,16.2,16.3,16.3,15.5,16.3,14.3,15.7,14.8,14.4,16.6,15.8,15,14.
T5:2016-01-01 00-00-00:rainfall:olympic:0,0,0,0,0,0,0,0,0,0,0,0.8,0.2,0.4,1,0,0,0,0,1,0.8,1.6,7,5.2,0.4,0,0,
T6:2016-01-01 00-00-00:rainfall:oakleigh:0,0,0,0,0,0,0,0,0,0,0.8,0,0.6,3.1,2.8,0,0,0,0,2,0.1,1.8,6.8,4.4,0,0
T7:2016-01-01 00-00-00:rainfall:moorabbin:0,0,0,0,0,0,0,0,?,0,0,0.4,0,0.8,1.8,0,0,0,0,1.8,1,1.8,4.6,1.8,0,0,
T8:2016-01-01 00-00-00:solar:olympic:22,25.1,25.7,15.3,25.8,18.1,30.4,25.4,21.3,31.3,15.8,29.1,25,3.9,19.7,3
T9:2016-01-01 00-00-00:solar:oakleigh:21.7,22.7,24.4,18.9,19.9,18.8,28.5,27,21.4,31.7,18.7,29.5,27.4,4,18.3,
T10:2016-01-01 00-00-00:solar:moorabbin:24.3,23.5,27,24.2,29.2,18.8,31.2,27.4,21.4,31.7,17.3,27.6,24.5,3.8,1
```

Figure 3: Weather data file generated by the provided script.

Important

The following dataframe shows the information that our R script, *write_tsf_weather_data.R* uses to download the weather data.

```
##        type  type_code p_nccObsCode      area station             p_c
## 1  max_temp IDCJAC0010          122   olympic   86338 <add_this_value>
## 2  max_temp IDCJAC0010          122 moorabbin   86077 <add_this_value>
## 3  min_temp IDCJAC0011          123   olympic   86338 <add_this_value>
## 4  min_temp IDCJAC0011          123 moorabbin   86077 <add_this_value>
## 5  rainfall IDCJAC0009          136   olympic   86338 <add_this_value>
## 6  rainfall IDCJAC0009          136  oakleigh   86088 <add_this_value>
## 7  rainfall IDCJAC0009          136 moorabbin   86077 <add_this_value>
## 8     solar IDCJAC0016          193   olympic   86338 <add_this_value>
## 9     solar IDCJAC0016          193  oakleigh   86088 <add_this_value>
## 10    solar IDCJAC0016          193 moorabbin   86077 <add_this_value>
```

In the above dataframe, the values corresponding with column "p_c" are tokens that are unique for a given weather information and a weather station. These tokens change everyday and hence, we cannot hardcode them in our R script.

Thus, before executing the *write_tsf_weather_data.R* script, make sure that you update the "p_c_values" variable defined in line 135 of the script with p_c values corresponding with the day you download the data. The "p_c_values" variable should be provided as a vector and it should contain the corresponding tokens in the same order as they appear in the above dataframe.

The following example shows how to identify the p_c value corresponding with maximum temperatures of the Olympic Park.

1. Go to http://www.bom.gov.au/climate/data/, select the type of weather data and enter the station number, and click on "Get Data", as shown in Figure 4.
2. Place the mouse pointer over "All years of data". You will see the downloading URL and the corresponding p_c value as shown in Figure 5.

You can follow the same procedure to identify the p_c values of other weather information corresponding with a given weather station. The station numbers are provided in the above dataframe.

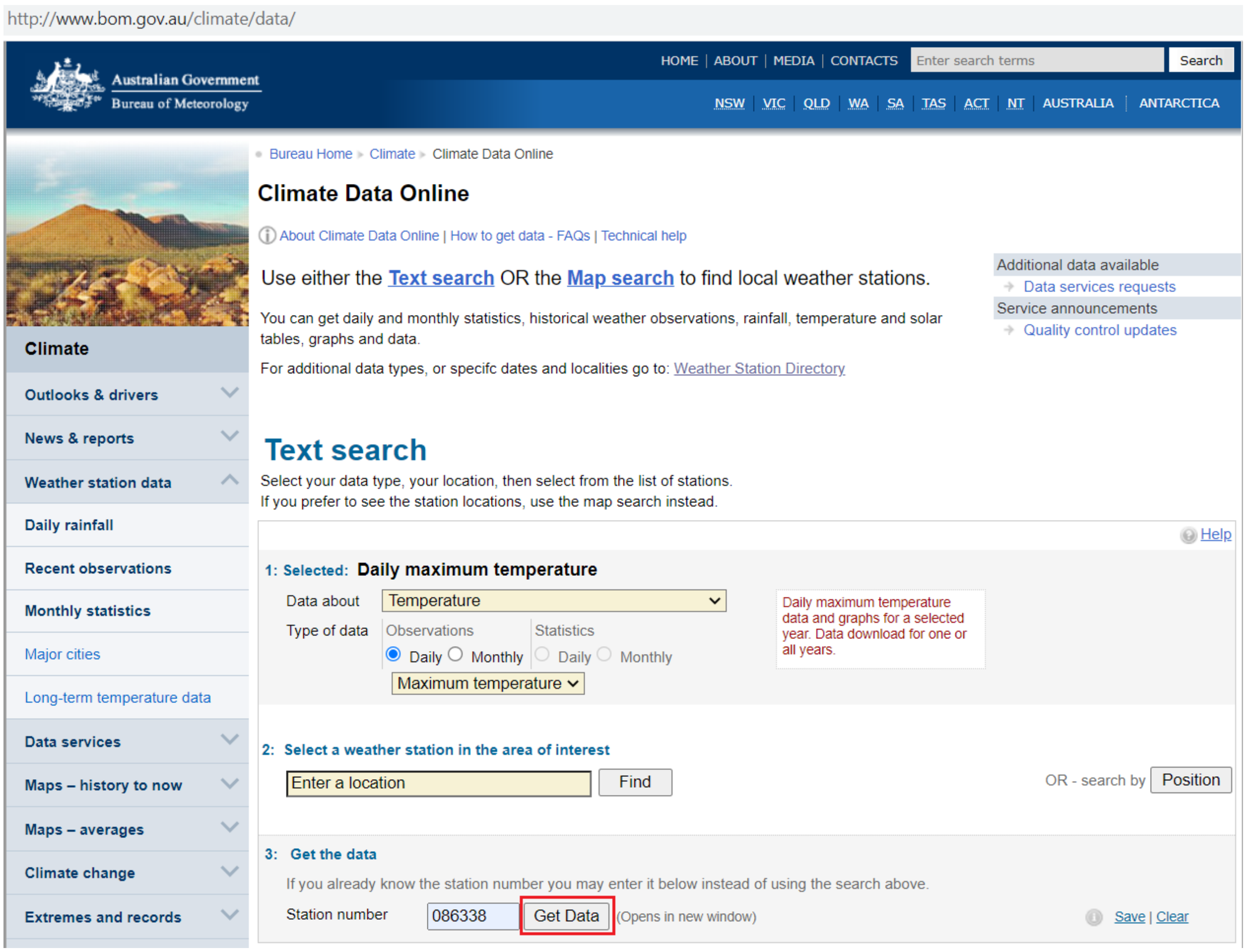


Figure 4: Extracting the identifiers needed for the script to download the weather data.

5 Optimisation Problem

We consider the problem of timetabling a set of activities for the coming month, across a set of buildings.

Each building has a base (background) load which follows the load profile provided in the dataset, and optionally a solar PV array. Building 0 has Solar 0, Building 1 has Solar 1, etc. Together, these give building i's net power draw at each time, $p_{i,t}$ (in kW).

Furthermore, some buildings have access to a battery, providing the capacity to store excess solar energy for later use. Batteries have a capacity, a maximum power and an efficiency rating. Batteries may also be charged from the grid, subject to the net load constraints.

Collectively, buildings place cumulative load $\bar{p}_t = \sum_i p_{i,t}$ on the grid. Cumulative load must always be positive, no feed-in, meaning $0 \leq \bar{p}_t$. The university is exposed to a peak load tariff, meaning that it must pay relative to the value of $\max_t \bar{p}_t$, taken over the whole month.

We are given a set of activities to schedule for the month. These activities include recurring activities (lectures scheduled at the same time each week), and once-off activities (experiments). Each activity has its own power draw p_{a_j} and duration d_j, resulting in a 'rectangular' load. Activities might have precedence constraints, such that $a_j \prec a_k$ means that a_j must happen *on a weekday before* a_k. In other words, if a_k is scheduled on a Wednesday, a_j can only be scheduled on Monday or Tuesday. In the case of recurring activities, such precedence constraints apply to each week (e.g., lecture before lab), whereas for once-off activities, the precedence constraints only need to be met across the whole month.

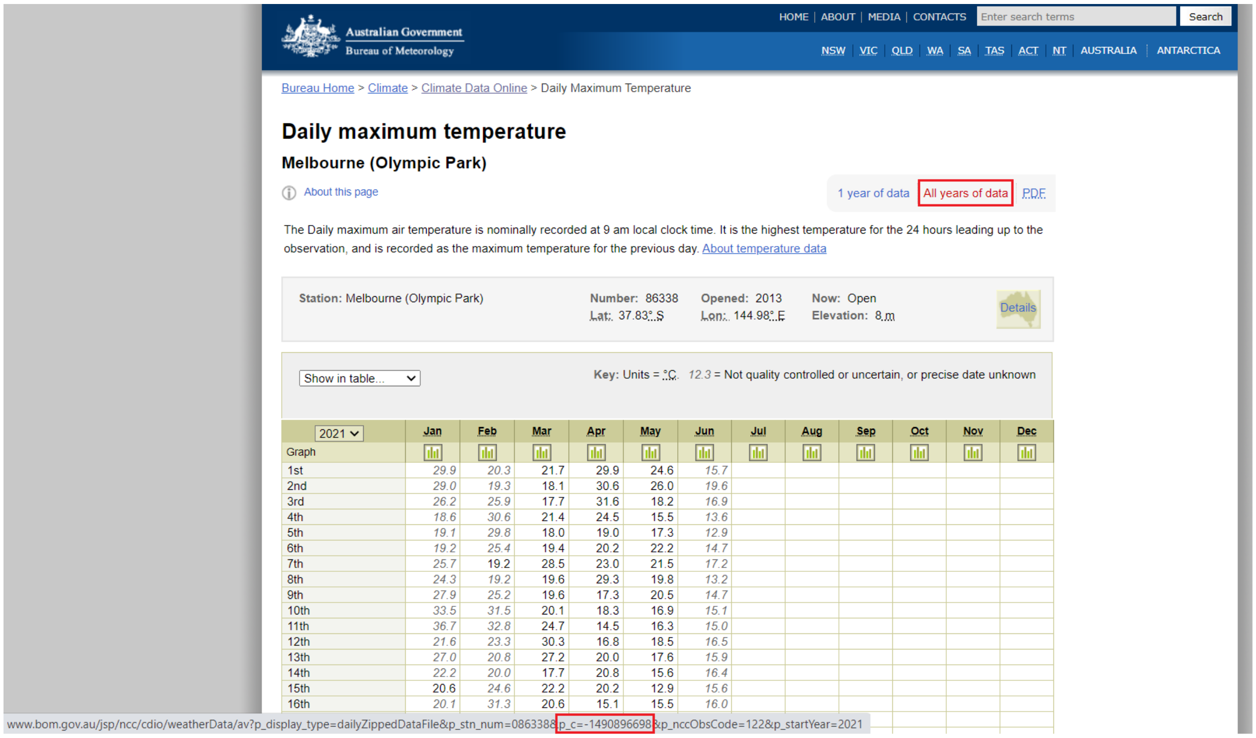


Figure 5: Extracting the identifiers needed for the script to download the weather data.

Recurring activities *must* all be scheduled, and always within office hours (starting on or after 9:00 and finishing before 17:00), whereas once-off activities can be dropped from the schedule. Each ‘deferrable’ once-off activity is worth a dollar value for performing the activity during office hours. Deferrable activities may also be scheduled after hours, which reduces its value.

Problem instance format

A problem instance begins with a line starting with ‘ppoi’ (predict-plus-optimize instance), followed by five numbers:

$$\texttt{ppoi}\ \langle \#\ \textit{buildings} \rangle\ \langle \#\ \textit{solar} \rangle\ \langle \#\ \textit{battery} \rangle\ \langle \#\ \textit{recurring} \rangle\ \langle \#\ \textit{once-off} \rangle$$

Each building, solar panel, battery, recurring activity and once-off activity is then given on its own subsequent line.

Building lines start with ‘b’ followed by the building identifier, which lines up with the buildings in the data set. The next two numbers following the identifier are for the dimensions of the building, the number of ‘large’ and ‘small’ rooms in the building.

$$\texttt{b}\ \langle \textit{building id} \rangle\ \langle \#\ \textit{small} \rangle\ \langle \#\ \textit{large} \rangle$$

Solar lines start with ‘s’, followed by two identifiers. The first identifier is the solar series in the data set, the second identifier is the building id to which the solar panel is attached.

$$\texttt{s}\ \langle \textit{solar id} \rangle\ \langle \textit{building id} \rangle$$

Battery lines start with ‘c’, followed by four numbers. The first number is the identifier of the building that the battery is attached to. Following that, we get the capacity of the battery in kWh and the maximum power charged to or discharged from the battery in kW. The third number gives the round-trip efficiency of the battery. The actual load the battery puts on the grid is modified by this efficiency factor, such that during charging it requires $max\ power \cdot \sqrt{efficiency}^{-1}$ kW, while discharging at $max\ power \cdot \sqrt{efficiency}$. Batteries start the month full.

$$\texttt{c}\ \langle building\ id \rangle\ \langle capacity\ kWh \rangle\ \langle max\ power\ kW \rangle\ \langle efficiency \rangle$$

Recurring activities start with 'r', followed by the identifier of the activity. Each activity then lists how many rooms it needs simultaneously, e.g., '3 L' for three large rooms, or '2 S' for two small rooms. After this room specification, the load in kW and the duration in 15-minute long time steps is given. The load is per room, so two rooms in the same building draw twice the kW value.

$$\texttt{r}\ \langle activity \rangle\ \langle precedences \rangle$$

Finally followed by the precedence constraints: one number listing the number of activities that must be scheduled before it, following by the ids of the (recurring) activities to be scheduled *on days earlier* in the week. All recurring activities must be scheduled within working hours (start on or after 9:00 and finish on or before 17:00 on a weekday).

$$\begin{aligned} \langle activity \rangle &= \langle act.\ id \rangle\ \langle \#\ rooms \rangle\ \langle \{S, L\}\ room\ size \rangle\ \langle load\ kW \rangle\ \langle duration \rangle \\ precedences &= \langle \#\ precedences \rangle\ \langle act.\ id \rangle^* \end{aligned}$$

Once-off activities start with 'a', but otherwise follow the spec of recurring activities. After the common components, once-off activities further have a dollar value associated with them, and a dollar value reduction if the activity happens to be scheduled (partially) outside working hours, such that an activity scheduled outside working hours receives \$ $(value - penalty)$.

$$\texttt{a}\ \langle activity \rangle\ \langle \$\ value \rangle\ \langle \$\ penalty \rangle\ \langle precedences \rangle$$

```
ppoi 3 2 1 4 2
b 0 1 2
b 1 1 0
b 2 0 1
s 0 0
s 1 2
c 0 5 2 0.87
r 0 1 L 15 8 1 2
r 1 2 S 8 12 0
r 2 2 L 10 4 0
r 3 1 S 4 4 0
a 0 2 S 8 12 500 100 0
a 1 2 L 8 16 2000 1500 1 0
```

Figure 6: Example instance format.

Figure 6 presents an example instance tying all the concepts together.

6 Submission

You are required to submit both forecasts and schedules corresponding with the test month (October 2020 for Phase 1 and November 2020 for Phase 2).

6.1 Submission of Forecasts

You must predict the energy demand/power production for all 12 series (6 buildings and 6 solar panels) for October 2020 (Phase 1) and November 2020 (Phase 2). The submission file should not contain a header. The forecasts of 12 series should be provided in 12 rows. The first column should display the series names as they appear in the *phase_1_data.tsf* file. The forecasts should be there from the second column onwards starting from the first 15-minutely period to the last 15-minutely period of the test month. Therefore, your submision should have 12 rows and <*forecast_horizon* + 1> columns.

Eg: for Phase 1, the submission file should contain 12 rows and 2977 columns.

You should save it as a **CSV file** and submit it together with the optimal schedules in a zip file as the **Analysis document**.

An example of a forecasts submission is shown in the file **sample_submission.csv**, as illustrated by Figure 7.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Building0	0	0	0	0	0	0	0	0	0	0	0	0
2	Building1	0	0	0	0	0	0	0	0	0	0	0	0
3	Building3	0	0	0	0	0	0	0	0	0	0	0	0
4	Building4	0	0	0	0	0	0	0	0	0	0	0	0
5	Building5	0	0	0	0	0	0	0	0	0	0	0	0
6	Building6	0	0	0	0	0	0	0	0	0	0	0	0
7	Solar0	0	0	0	0	0	0	0	0	0	0	0	0
8	Solar1	0	0	0	0	0	0	0	0	0	0	0	0
9	Solar2	0	0	0	0	0	0	0	0	0	0	0	0
10	Solar3	0	0	0	0	0	0	0	0	0	0	0	0
11	Solar4	0	0	0	0	0	0	0	0	0	0	0	0
12	Solar5	0	0	0	0	0	0	0	0	0	0	0	0
13													

Figure 7: Sample submission of the forecasts.

6.2 Submission of Schedules

For the optimization side of the problem, you are expected to provide one schedule for every instance in the test set. Minimally viable baseline solution schedules are provided as part of the instance pack, which both act as a template for your solutions and as a starting point if you wish to focus only on one aspect (e.g., battery). They can also be used as default answer when particular instances turn out to be unsolvable within reasonable time.

Schedule solution format

A solution to an individual instance takes the form of a plain-text (.TXT) file, with a similar grammar to the instance format.

The first line identifies the instance, by repeating the ‘ppoi’ header of the instance being solved. This line is followed by the schedule header, starting with ‘sched’, followed by the number of scheduled recurring activities and the number of once-off activities:

$$\texttt{sched}\ \langle \#\ \textit{recurring scheduled} \rangle\ \langle \#\ \textit{once-off scheduled} \rangle$$

The starting time of each *scheduled* activity is then specified on its own subsequent line. Finally, the battery schedule is given by listing the periods in which the battery is charging and discharging.

Scheduled activities start with either ‘r’ for recurring, or ‘a’ for once-off, followed by the identifier of the activity. Each activity then lists the period in which it starts, followed by a list of buildings occupied, one for each room the activity needs.

$$\langle \{r, a\}\ \textit{type} \rangle\ \langle \textit{act. id} \rangle\ \langle \textit{start time} \rangle\ \langle \#\ \textit{rooms} \rangle\ \langle \textit{list of building IDs} \rangle$$

Following the scheduled activities, the battery schedule is provided by one line per time step. Each line starts with 'c' for the battery, followed by the ID of the battery, the time period, and the decision to charge, discharge or hold charge.

$$\texttt{c}\ \langle \textit{battery id} \rangle\ \langle \textit{time} \rangle\ \langle \{0, 1, 2\}\ \textit{decision 0=charge, 1=hold, 2=discharge} \rangle$$

Decisions to hold charge may be omitted as holding charge is the default action, meaning that a valid battery schedule could have only lines with decision 0 or 2.

```
ppoi 3 2 1 4 2
sched 4 2
r 0 4 1 0
r 1 4 2 0 0
r 2 0 2 0 0
r 3 0 1 0
a 0 16 2 0 0
a 1 28 2 0 0
c 0 0 2
c 0 1 2
c 0 10 0
c 0 11 0
c 0 24 2
...
```

Figure 8: Truncated solution example demonstrating the solution format.

Figure 8 presents an example solution tying all the concepts together.

6.3 Other submission requirements

Important: In addition to their predictions and schedule, participants are requested to submit a draft description their methodology **in their final submission** (up to 1000 words). This is compulsory as this preliminary description will also be evaluated by the committee to shortlist the top 5 submissions. Please use the 'Abstract' field to include this information when submitting your final analysis. If your submission is shortlisted you will be asked to provide a final description within 4 weeks after the final deadline.

Detailed instructions to register and submit your solutions can be found here.