Short-term scheduling simulations

Aleksandar Cikota (ESO)

---

1. Introduction

The purpose of the short-term scheduling simulations is to investigate the effect of different filtering and ranking constraints and conditions (e.g. the seeing predictions, change of instrument condition, wind speed pointing and dome closure constraints, etc.) on the operations efficiency (e.g. by measuring the cumulative time of Must-repeat OBs, or the cumulative completion rate), with the main goal to improve the efficiency of the Paranal operations.
The scripts and simulations in this work are pilot study, focused mainly on investigating the effect of different seeing prediction models (precast10min, deep learning forecast, and random forest forecast), applied to UT1 (FORS2, KMOS and ESPRESSO) during the semester of April-September 2021.

2. Simulation script

Necessary input files:

- A list of Observing blocks (OBs): SMTSqueues/queue_dump_2021-04-01.csv or SMTSqueues/queue_dump_2021+2022-04-01.csv, which are lists of OBs at the beginning of the April 2021 semester, and the merged lists of April 2021 and April 2022, respectively.
- The schedule of technical or VLTI nights, P107schedule.txt, in which there are normal nights (indicated 'Normal') or technical/VLTI nights (indicated '1'). This reduces the number of nights during which we simulate observations. I indicate all VLTI nights as '1', which reduced the number of nights from 183 to 134.
- Real DIMM information: weather/wdb_query_1980_eso.csv, which is used to calculate the precast10min, and for grading of OBs
- Sky transparency information: weather/weatherlog-VISTA-clean.csv

Seeing prediction models:

- Nowcast60: seeing predictions based on deep learning (Fuyan_nowcast/nowcast_pred_1h.csv)
- RF60: seeing predictions based on random forest (Angel_nowcast/PAO_2021_seeing_nowcast_RF_scheduling_simulations_20220319.csv)
- RF60mod: Nathalie's Improved Seeing predictions based on random forest (RFmod_v9_22.csv)
- Precast10min: The median DMM seeing calculated from last 10min before the OB execution

The simulations script works, in summary, as follows:

1.) For each night, starting from April 1, 2021 (until September 30, 2021), the twilight times (beginning and end of night) are calculated. At the beginning of night, current_time is defined as the time of the end of twilight. The default current instrument at the beginning of night is FORS2.
2.) While current_time < time of the end of night, the simulation is running as follows
3.) The current DIMM seeing is taken from the seeing prediction models (see section 'Seeing prediction models' above).
4.) The current sky transparency is taken from the sky transparency database

5.) Script performs check whether there are real seeing information available for the current time. If DIMM information is not available (which is not very often the case), then the simulation goes IDLE for 1h and current_time=current_tme+1h. Note that DIMM is necessary for grading of the OBs.
6.) Ranking of OBs based on the following input parameters: date of night, current time, current transparency, current seeing. For the detailed ranking procedure, see Section 2.1 'Filtering and Ranking'.
7.) Ignore ESPRESSO targets from the list of ranked OBs that have been observed during the same night (that is a simplified way to deal the ESPRESSO time links).
8.) If the list of ranked OBs is empty, simulation goes IDLE for 10min.
9.) Change of instrument if the difference in ranking of the highest ranked OB is >0.2 compared to the highest ranked OB for the current_instrument. The default change of instrument requirement of 0.2 can be modified (which we also did in some simulations, to investigate the differences in the efficiency using different change of instrument conditions). Change of the instrument takes 7 minutes.
10.) Grade the OB on behalf of the real DIMM seeing. Because the 'seeing' requirement in the OBs is actually image quality (IQ), the DIMM seeing is recalculated to IQ before grading. See section 2.2 'DIMM2IQ conversion and grading' for more information. The OBstatus changes accordingly.
11.) Check if the transparency at the end of OB is still fulfilled. If the transparency changed, we grade as C (must repeat). This doesn't happen often. The OBstatus changes accordingly.

An example of one night is shown in Figure 1.

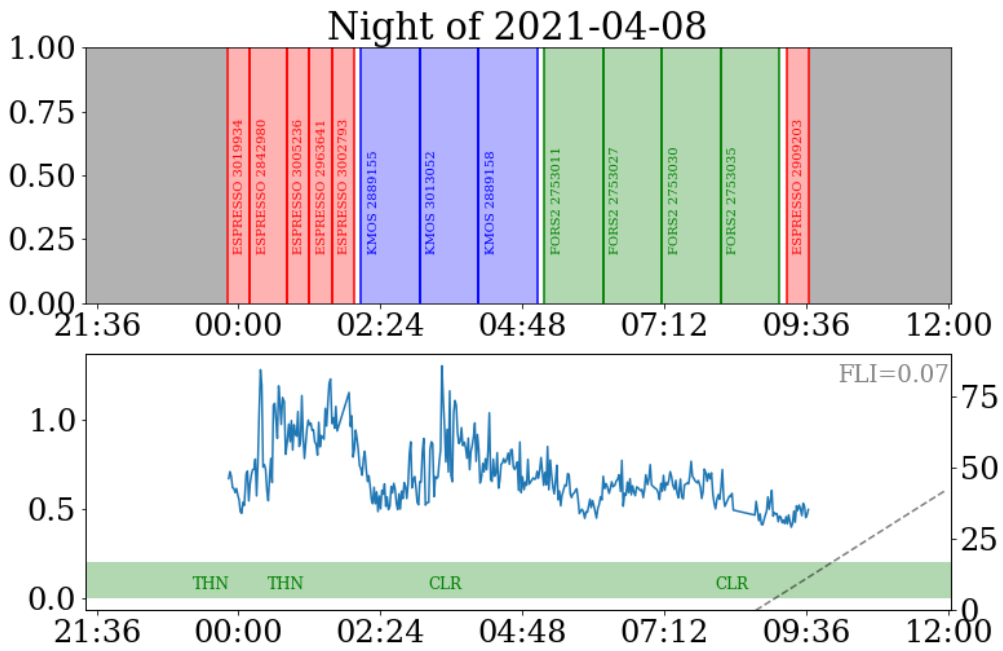

Figure 1: An example of 1 simulated night. The top panel shows the executed simulated OBs, and the bottom panel shows the DIMM, sky transparency conditions and the Moon phase and altitude (dashed line) during the night.

2.1. Filtering and ranking

The ranking works as follows, step-by-step:

1.) We calculate the Moon phase and position at the current time
2.) We calculate current altitudes and airmasses for all OBs
3.) Filtering: We take into account all OBs which fulfill the following conditions: currentAltitude>0, and (OBstatus=='+' or OBstatus=='M') and (instrument=='FORS2' or instrument=='KMOS' or instrument=='ESPRESSO')
4.) Transparency filtering: check whether the OBs fulfill the current_transparency condition
5.) I calculate the Moon distances for all previously filtered OBs
6.) Filtering: We take into account all OBs that fulfill the Moon distance, Moon phase and airmass requirements.
7.) We calculate the IQ requirements from the current DIMM for all the previously filtered OBs
8.) Filtering: We take into account all OBs which fulfill the seeing (i.e. IQ) requirement.
9.) We check whether the ending time of the OB is before the start of morning twilight. If not, the OB gets skipped until a OB with a shorter-duration is found.
10.) Ranking: Following the Ot-survey-support_2021.pdf manual, the ranking is based on the probability: rank= Psky(current transparency) * Pz(airmass, dec) * Pset(airmass, ra, dec, date of night)* Pseeing(current seeing)* Pfli(current_FLI)
(see Ot-survey-support_2021.pdf for details)
11.) If the OB is B ranked by the OPC, I add +0.5 to the rank, and +1.0 in case of C-ranked OBs.

An example of the output of the ranking function is shown in Figure 2.

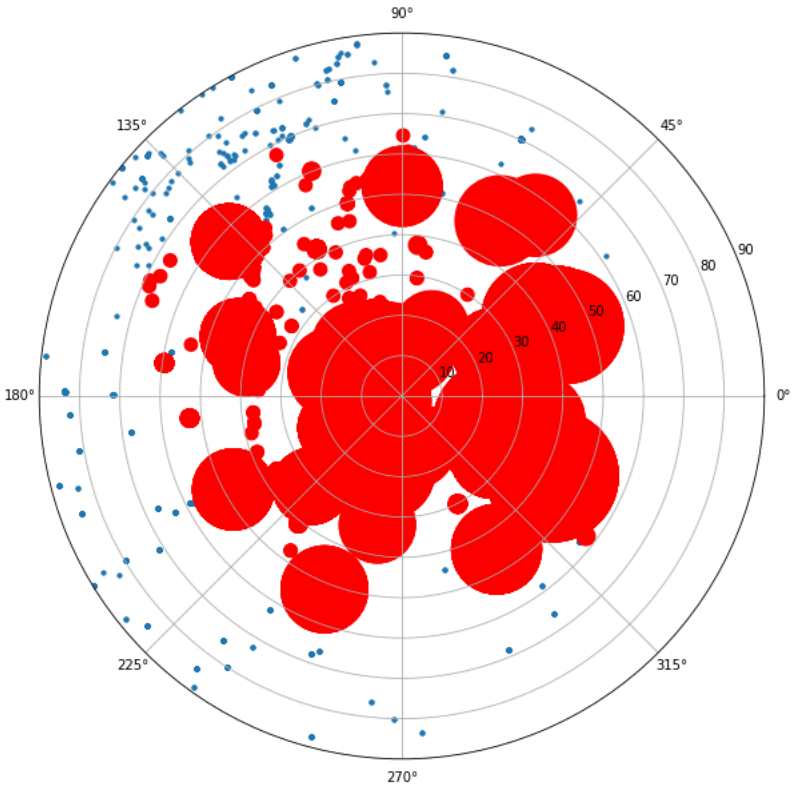

Figure 2: The blue dots are all targets above the horizon at a given time, and the red dots are the ranked targets. The bigger the dot is, the lower the rank is (the targets with smallest rank value will be observed).

2.2. DIMM2IQ conversion and grading

To grade the OBs, we convert the DIMM seeing to image quality, because the 'seeing' requirement in the OBs are actually image quality. Following the Ot-survey-support_2021.pdf manual, the conversion is done as function of the input DIMM, instrument, instrument filter, and the airmass.

```
def FWHMIQ(FWHM_ORANG, instrument, insfilter, airmass):
  f=open('ins_filt_lc_fwhm.txt')
  for line in f:
    if instrument in line:
      filt=line.split()[1]
      if insfilter==filt:
        lc=float(line.split()[2])
        FWHM_INS=float(line.split()[3])
       # print (filt, lc, FWHM_INS)
  f.close()

  lc=lc*1e3
  L_0 = 46 #m
  F_Kolb = -0.982  ## VLT
  AO_GAIN = 1.0
  r_0 = 0.976 * 500.0e-9/FWHM_ORANG*((180/np.pi)*3600)* (lc/500)**1.2 * (airmass**(-3/5))

  FWHM_ATM = FWHM_ORANG * (lc/500)**(-1/5) * airmass**(3/5) * (1 + F_Kolb * 2.182 * (r_0/L_0)**0.356)**0.5

  if (1+F_Kolb * 2.182 * (r_0/L_0)**0.356) < 0:
    FWHM_ATM = 0
  FWHM_TEL = 0.000212 * lc / 8.2  ## VLT
  FWHM_IQ =  ((FWHM_ATM/AO_GAIN)**2.0 + FWHM_TEL**2.0 + FWHM_INS**2.0)**0.5
  return FWHM_IQ
```

There are two grading schemas that are currently implemented and can be used in the simulation.

i) 10% of time above 10% of requirement
   - If the seeing (i.e. IQ) is less than 10% of the execution time of the OB above the A grade requirement, the OB gets graded as 'A'
   - If the IQ is less than 10% of the execution time of the OB above the B grade requirement (which is IQ*1.1), the OB gets graded as 'B'
   - If the IQ is more than 10% of the execution time above the B grade requirement (which is IQ*1.1), the OB gets graded as 'C'

ii) Average IQ
   - If the averageIQ of the OB is less than the IQ requirement, the OB gets graded as 'A'
   - If the averageIQ of the OB is larger than the than the IQ requirement, but smaller than the IQrequirement*1.1, the OB gets graded as 'B'
   - If the averageIQ of the OB is larger than the IQrequirement*1.1, the OB gets graded as 'C'

Figure 3 shows an example of the seeing analysis function for one OB. starting on '2021-05-18T23:25:53.679' and ending on '2021-05-19T00:35:00.680' for a target observed at airmass 1.482, with ESPRESSO. Note that on the y axis the IQ is plotted (and not DIMM as indicated). The DIMM has been recalculated to IQ using the FWHMIQ function (See Sect. 2.2). The yellow line shows the IQ 10min before the beginning of the OB, with the median (horizontal line). The red horizontal line indicates average of the IQ, and the blue horizontal line indicates the IQ requirement for that specific OB. The green dots above the blue line indicate the IQ points between the IQ requirement and the IQ*1.1, while the red dots indicate the IQ points above the B grade requirement (which is IQ*1.1).

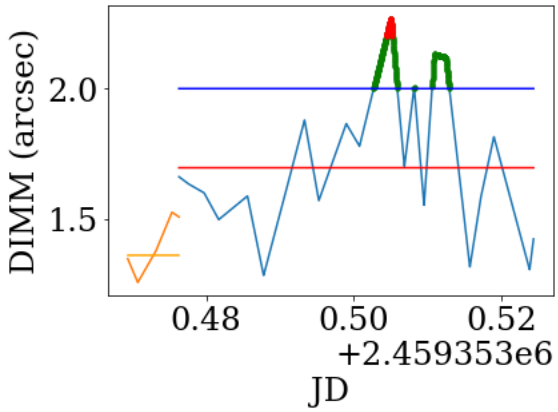

Figure 3: Image quality analysis.

2.3. Simulations output

The main output of the simulations are observing_log*.txt files, which contain the following columns:

- ob_id
- OB_start_jd
- OB_end_jd execution_time_sec
- status (e.g. 'C'ompleted, or Must-repeat, or '+')
- grade_of_OB
- IQ_requirement
- Instrument
- OBAirmassAtEnd
- aboveAreqdurationfrac (fraction of time with IQ above the A-grade requirement)
- aboveBreqdurationfrac (fraction of time with IQ above the B-grade requirement)
- average_IQ (average IQ over the total duration of the OB)
- precast10min (median of DIMM in arcsec)
- rank_class_of_OB
- [and some contain also precast10minIQ (median of IQ)].

Furthermore, IDLE times, change of focus, and IDLE due to missing seeing information are also logged in the observing log.

3. Pilot simulation runs and results

We ran 13 simulations which can be grouped in 4 sets of simulations.

- Set 1 investigates differences between the 3 different seeing prediction models using an oversized queue which consists of April 2021+2022 queues combined
- Set 2 investigates differences between the 3 different seeing prediction models using the April 2021 queue. This set is the only one which has an additional (fourth) simulation with an updated Random Forrest seeing prediction model.
- Set 3 investigates differences between the 3 different seeing prediction models using the April 2021 queue, however, the grading is based on the average IQ (while in Set 1, 2, 4 and 5 the grading is done with the 10% of 10% scheme)
- Set is the same as Set 2, but the instrument changes immediately, whichever instrument is required for the highest ranked OB.

Set 1	Python script: scheduling_simulation_v2.ipynb Input queue: queue_dump_2021+2022-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: precast10min Output log file: observing_log_precast.txt Night-plots folder: output_precast
	Python script: scheduling_simulation_v2-DL.ipynb Input queue: queue_dump_2021+2022-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: nowcast_pred_1h.csv Output log file: observing_log_DL.txt Night-plots folder: output_DL
	Python script: scheduling_simulation_v2-RF.ipynb Input queue: queue_dump_2021+2022-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: Angel_nowcast/PAO_2021_seeing_nowcast_RF_scheduling_simulations_20220319.csv Output log file: observing_log_RF.txt Night-plots folder: output_RF
Set 2	Python script: scheduling_simulation_v2-copy.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: precast10min Output log file: observing_log_precast_2021.txt Night-plots folder: output_precast _2021
	Python script: scheduling_simulation_v2-DL-copy.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: nowcast_pred_1h.csv Output log file: observing_log_DL_2021.txt Night-plots folder: output_DL_2021
	Python script: scheduling_simulation_v2-RF-copy.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS

Set 2 (cont.)	Instrument change if rank difference >0.2 Seeing model: Angel_nowcast/PAO_2021_seeing_nowcast_RF_scheduling_simulations_20220319.csv Output log file: observing_log_RF_2021.txt Night-plots folder: output_RF_2021
	Python script: scheduling_simulation_v2-RFmod.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: Angel_nowcast/RFmod_v9_22.csv Output log file: observing_log_RFmod_2021.txt Night-plots folder: output_RFmod_2021
Set 3	Python script: scheduling_simulation_v3.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: average IQ Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: precast10min Output log file: observing_log_precast_avggr.txt Night-plots folder: output_precast_avggr
	Python script: scheduling_simulation_v3-DL.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: average IQ Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: nowcast_pred_1h.csv Output log file: observing_log_DL_avggr.txt Night-plots folder: output_DL_avggr
	Python script: scheduling_simulation_v3-RF.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: average IQ Instruments: ESPRESSO, FORS2, KMOS Instrument change if rank difference >0.2 Seeing model: Angel_nowcast/PAO_2021_seeing_nowcast_RF_scheduling_simulations_20220319.csv Output log file: observing_log_RR_avggr.txt Night-plots folder: output_RF_avggr
Set 4	Python script: scheduling_simulation_v2-copy2.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change immediately (whatever instrument highest ranked OB requires) Seeing model: precast10min Output log file: observing_log_precast_inst.txt Night-plots folder: output_precast_inst
	Python script: scheduling_simulation_v2-DL-copy2.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change if immediately (whatever instrument highest ranked OB requires) Seeing model: nowcast_pred_1h.csv Output log file: observing_log_DL_inst.txt Night-plots folder: output_DL_inst
	Python script: scheduling_simulation_v2-RF-copy2.ipynb Input queue: queue_dump_2021-04-01.csv OB Grading: 10% of 10% Instruments: ESPRESSO, FORS2, KMOS Instrument change immediately (whatever instrument highest ranked OB requires) Seeing model: Angel_nowcast/PAO_2021_seeing_nowcast_RF_scheduling_simulations_20220319.csv Output log file: observing_log_RF_inst.txt Night-plots folder: output_RF_inst

Figure 4 shows summary plots for the 13 simulations. The cumulative time of completed and must repeat OBs at the end of the simulation (183 days after the beginning) are given in Table 1.

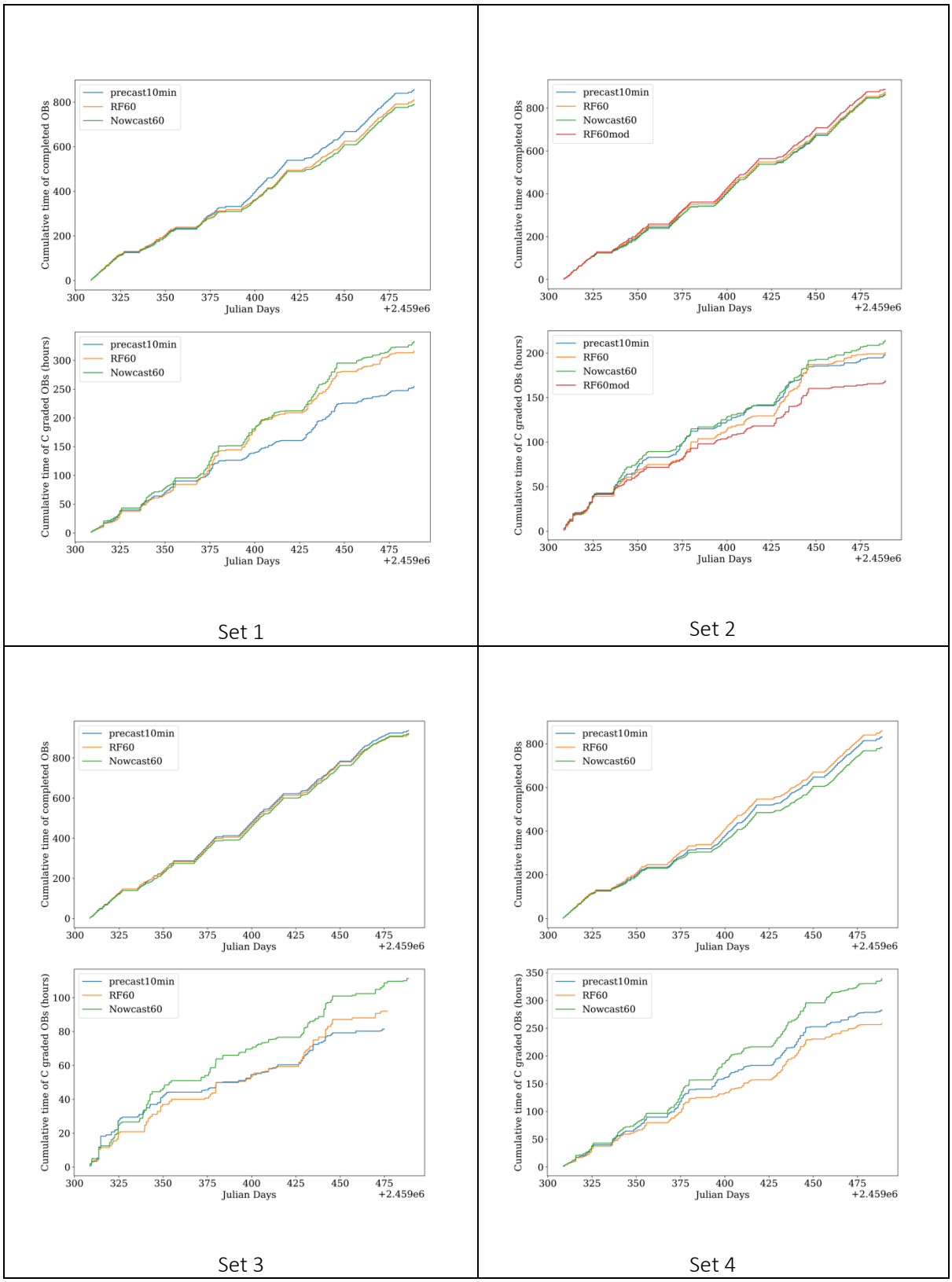

Figure 4 shows summary plots for the 13 simulations.

	Model	Cumulative time of completed OBs (h)	Cumulative time of C graded OBs (h)
Set 1	precast10min	856.3	254.7
	RF60	810.7	316.2
	nowcast60	790.8	333.0
Set 2	precast10min	866.4	198.8
	RF60	875.9	200.7
	RF60mod	887.9	169.1
	nowcast60	863.0	214.4
Set 3	precast10min	937.1	81.5
	RF60	917.8	92.0
	nowcast60	922.4	111.3
Set 4	precast10min	833.6	283.2
	RF60	860.7	258.8
	nowcast60	783.5	339.6

Table 1: Cumulative time of completed and Must-repeat OBs for 13 simulations.

4. Summary and conclusion

In this pilot study we ran short-term scheduling simulations for 1 semester on UT1 with the main goal to investigate the effect and benefits of different seeing forecast models. We also demonstrate the potential use of the simulations for the purpose of the optimization of the observing strategy, e.g. determining of the best change of instrument condition, and for optimizing the grading scheme.
Specifically, in regard of the seeing forecast models, we show that the most 'RFmod' shows promising results, compared to the other seeing forecast models (see Set 2 in Figure 4 and Table 1).
We also show the differences between different grading schemes (see Set 2 vs. Set 3). Simulations in Set 3, which have a less strict grading scheme achieve better efficiency (i.e. higher cumulative completion rates and lower cumulative time of C-graded OBs). Although it may be good for the efficiency, this may in return have a negative effect on the scientific outcome of the observations due to lower image quality of the observations.
Furthermore, we also demonstrate the use of the simulations to for the purpose of optimizing the observing strategy, for example, to determine the change of instrument condition. We show that it is not efficient to change the instrument immediately, whenever the highest ranked OB requires a different instrument (see Set 2 vs Set 4).

Although the current differences between the seeing forecast models are relatively small, we believe that there is a high potential for further improvements of the forecast models in the future (see e.g. differences between RF60 and RF60mod in Set 2). Furthermore, we also demonstrated that running such simulations may be beneficial in the future for the optimization of operations and testing of different seeing forecast (and in general weather forecast) models.