gWTO2 Documentation (Draft)

Release 2.0RC

itoledo@alma.cl

July 28, 2014

Download documentation in PDF.

CHAPTER

ONE

INTRODUCTION

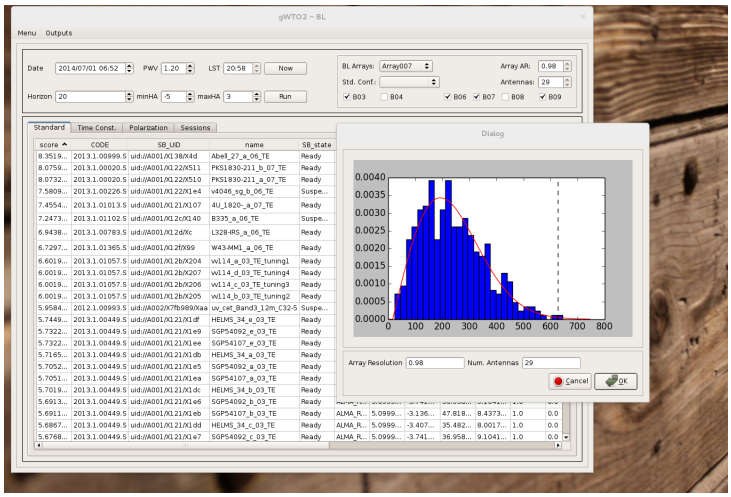

Figure 1.1: Screenshot of gWTO2 RC

1.1 Goal

gWTO2 (gui for What To Observe cycle 2) is a real-time scheduler that uses both a selection and score algorithms to select the best SB to observe given the current conditions. However, its main goal is to provide a flexible and dynamic benchmark where to test and optimize the algorithms that are being use in the official scheduling tool for ALMA: the Dynamical Schedulling Algorithm (DSA).

The criteria behind gWTO2 comes from Alma Cycle 2 Proposer Guide (http://almascience.eso.org/documents-and-tools/cycle-2/alma-proposers-guide):

> Science observations will be executed by ALMA operations staff, taking into account (in rough order of priority): the weather conditions, the configuration of the array, target elevation and other practical constraints, the projects' assigned priority group, and executive balance. All other things being equal, the project with the highest scientific rank will be observed."
>
> —*Section 5.1 of the ALMA Cycle 2 Proposers Guide.*

1.2 Considerations and problem description

It is easy at any given time to use these criteria to create an algorithm that selects the best SB to run, given the current pwv, array configuration, number of antennas and the target's horizontal coordinates; however the algorithm needs to be more complex to achieve an optimal efficiency in the use of telescope time, which finally traduces in a maximum scientific output.

CHAPTER

TWO

USING GWTO2

2.1 Starting the GUI

gWTO2 is tested and deployed at the **osf-red** machine, within the **aod** account. A virtual environment of python, based on the Anaconda distribution (http://docs.continuum.io/anaconda/index.html), must be loaded before using it. This is achieved by running:

```
. activateC2Test
```

The Anaconda distribution is based on python 2.7.6 and includes numpy, pandas, pyephem and other libraries need by gWTO.

The gui is run executing the gWTO2.py comand:

```
Usage: gWTO2.py arg1 [options]
    arg1 must be BL or ACA

Options:
  -h, --help            show this help message and exit
  -c, --clean           Force clean up of gWTO2 cache
  -p PATH, --path=PATH  Path for cache
```

So, to run gWTO2 for baseline correlator use argument *BL*, and *ACA* for Total Power and ACA. The *-c* option should only be used once per day.

I would also recommend to set the option *-p* to something like `'/.wto_myname/'` so different users running gWTO won't mess up the cache for each other. **After playing with gWTO2 using a different path, please delete the directory created with the name .wto_myname**

2.2 The gWTO2 window

After starting **gWTO2 BL** you will be presented with the gui shown on *Figure 2* (page 6). If the *-c* option was used, or the cache have been manually erased it, the time until the gui is ready can be up to 5-7 minutes.

The gui for ACA (**gWTO2 ACA**) is almost the same, except for the lack of *Array Options*, and the presence of a tab *TP* that will be used for handling Total Power SBs.

2.3 Setting up variables

After opening, the *Date* will be by default the current UTC time, *PWV* is set to 1.2 mm., the *Horizon* limit is 20 degrees, *minHA*, minimum hour angle, is -5 and *maxHA*, maximum hour angle, is 3. The *LST* field is not editable, and it shows the LST for the date/time set in the *Date* field.

(For BL GUI only.) The box with the array variables will have the *Std. Conf.:* field set to `Current Conf.` This `Current Conf.` comes from the output of the CASA script **arrayConfigurationTools.py**, which can be found at

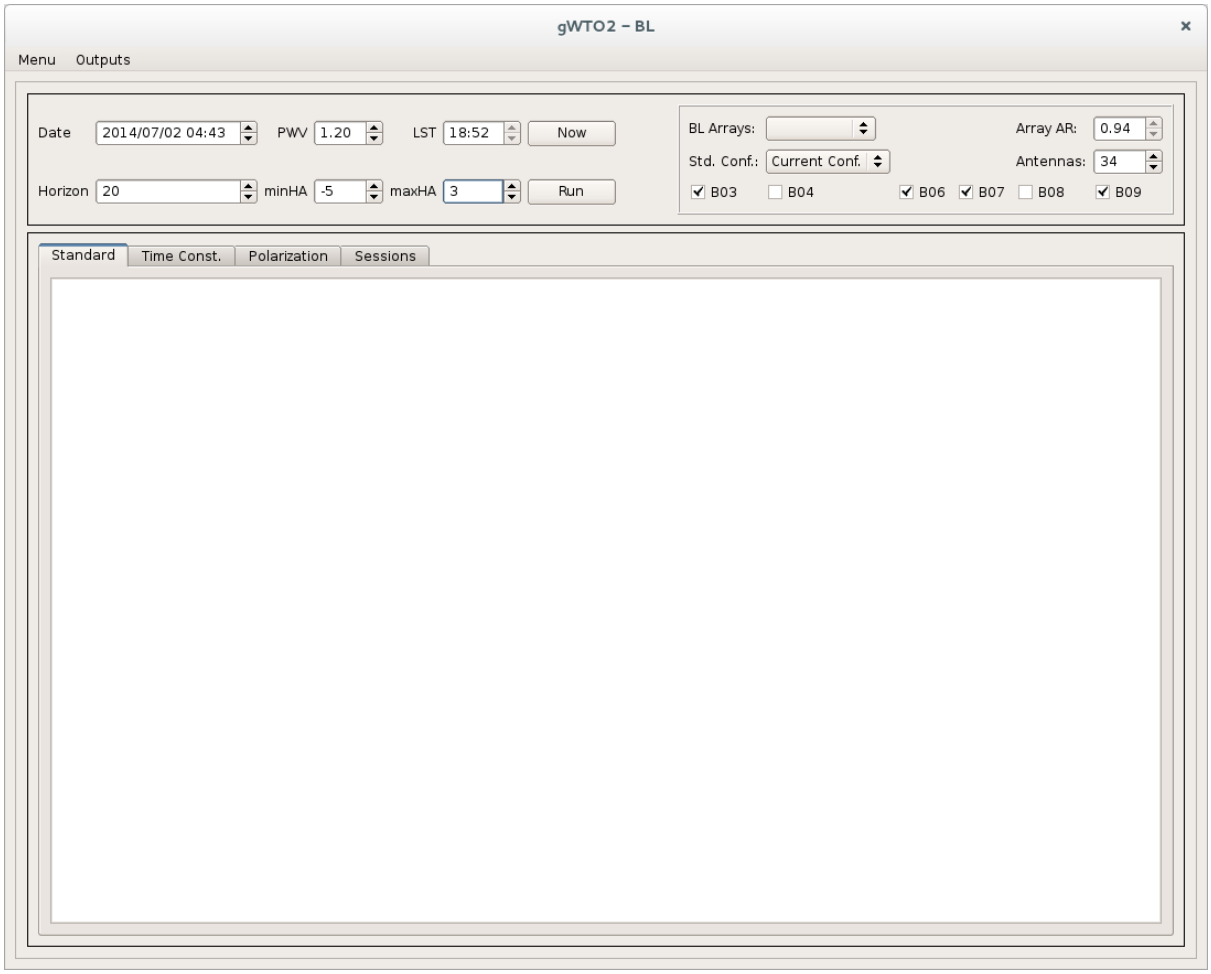

Figure 2.1: Figure 2

`~/AIV/science/ArrayConfiguration/Tools/arrayConfigurationTools.py`. It is made with the antennas that in principle can be used for the current ES Block. **It is the AoD Lead's duty to create the relevant files from time to time to account for antenna movements or new antennas added**. *(Instructions)* (page 31)

(For BL GUI only.) The values given at *Array AR:* and *Antennas:* are set according to the *current array's* angular resolution and number of antennas offered officially for cycle 2. **The only field you can modify at this stage in the 'Antenna' field, which is the number of antennas**. The idea is that the user will use this information to have an idea of the current configuration characteristics, and to run gWTO2 to plan observations ahead of time, or when Baseline arrays have not been created in the last 6 to 12 hours.

(For BL GUI only, when observing.) The user should press the button *Now*, and a pop up window similar to the one shown in *Figure 3* (page 7) will appear.

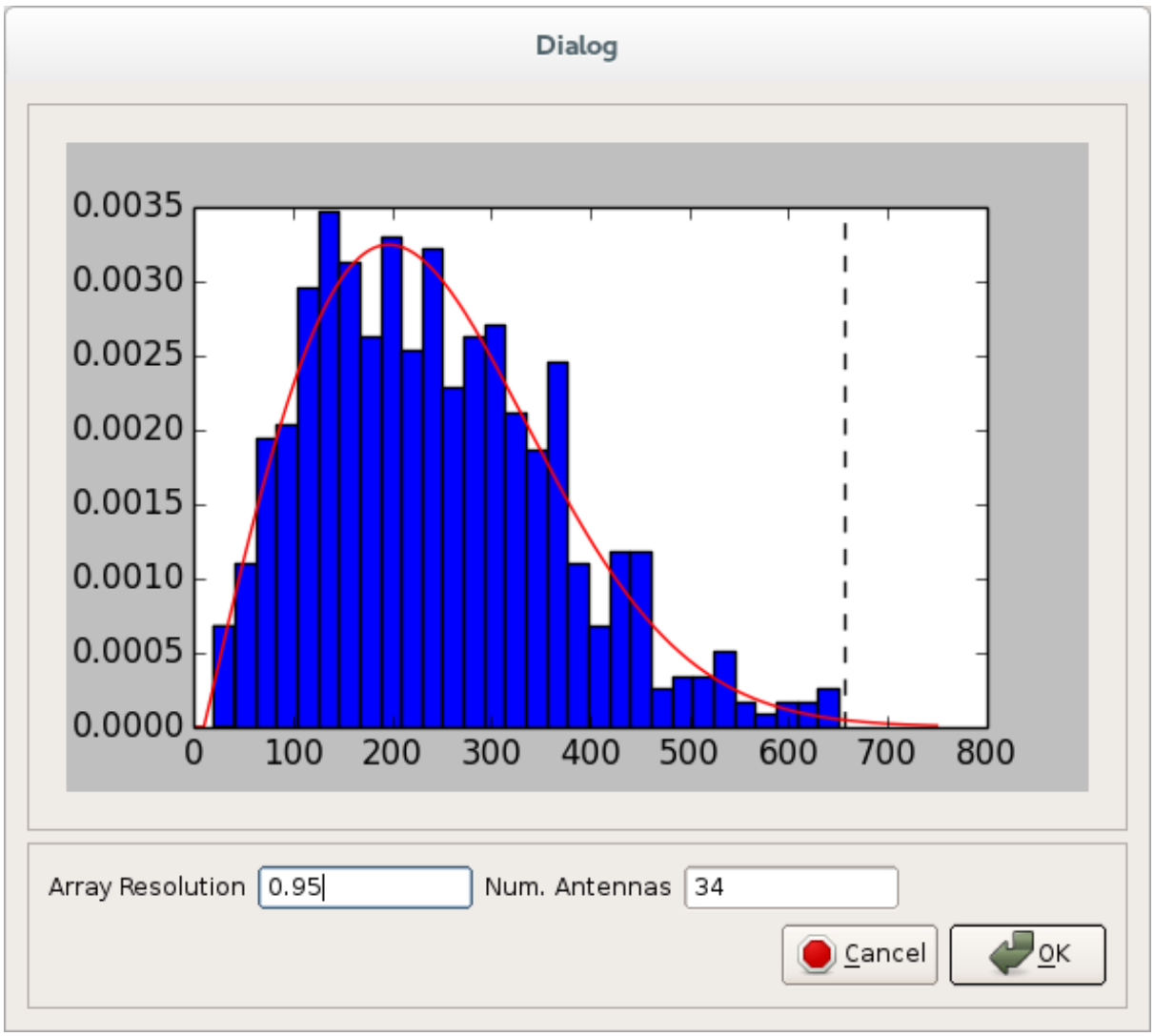

Figure 2.2: Figure 3

The window show the normalized histogram of the baseline lengths, and a fit to this distribution, taking the data from latest Baseline Array created. From this distribution the array's resolution is estimated, and the number of antennas is also show. **The user should check that the array resolution is close the the "Current Conf." value, and that no outliers are fitted**. If happy press the *OK* button, and this will set the *Array AR:* and *Antennas:* fields in the main window. If *Cancel* is pressed instead, the main window will go back to `Current Conf.`. Also, when accepting the new array estimates you will not longer be able to change the number of antennas unless you go back to :keyword"*Current Conf.*

The *BL Arrays:* Combo menu is also populated with the list of the baselines arrays created in the last 6 to 12 hours.

For ACA GUI only The number of antennas is 9 by default. Change the number according to the number of antennas that are available.

2.4 Running

When you are happy with the *Date*, *PWV* and array variables (also the *Horizon*, *minHA* and *maxHA* values) you can run the selector and scoring algorithms pressing the button *Run*.

After an interval of a few seconds (5 to 15 seconds) you will be presented with something similar to *Figure 4* (page 8).

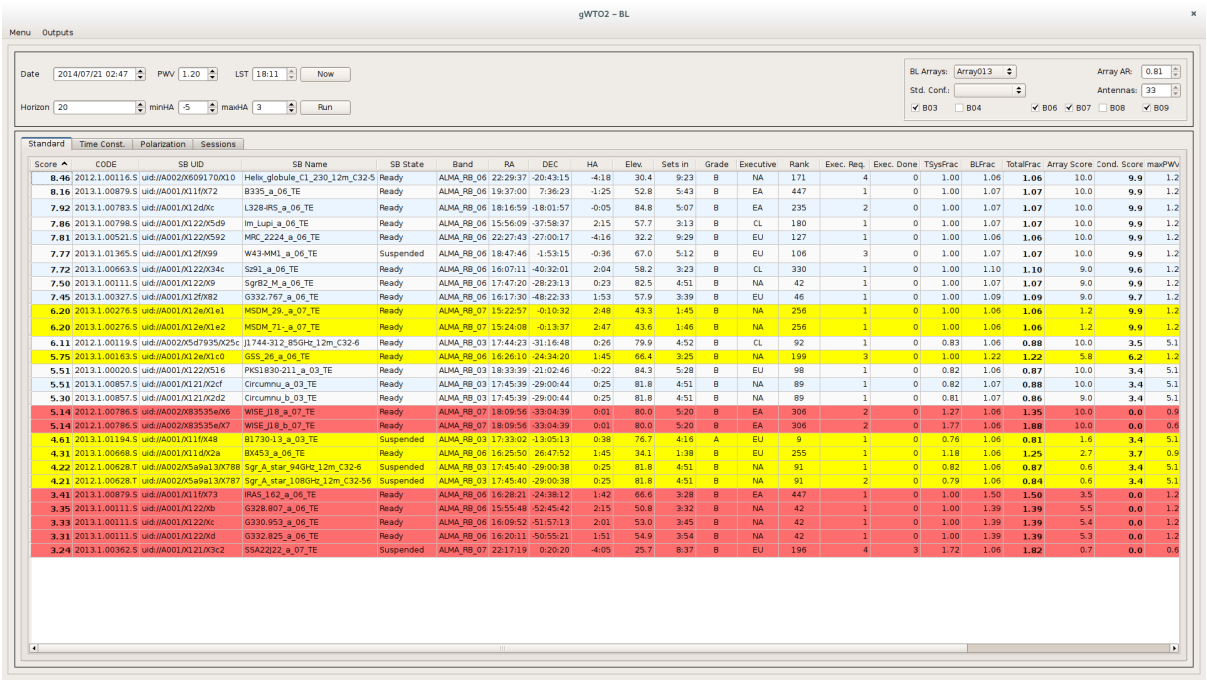

Figure 2.3: Figure 4

2.5 Reading the output Scores

2.5.1 Meaning of the background colors

- **Red**: Means that the integration time for the science target needs to be increased by over a factor of 1.3 to reach the PI requested sensitivity. **Do not change the integration time unless a clear policy has been set. (AoD leader should know.)**
- **Yellow**: The SB, even when it can be run with the current Array's angular resolution, does require, or prefers, another configuration. If there is nothing else to observe, the AoD could execute it. **This doesn't apply to point sources.**

2.5.2 Standard (ACA) Tab in BL (ACA) GUI

1. **Score:** The score is the weighted mean of different scores calculated for each observable SB. The score is a value between 0 and 10, 10 being the highest score.

(a) Condition Score, 35%. A score depending on the current PWV, number of available antennas, and pwv used by the OT.

(b) Array Score, 20%. Depends in how close to the current array's resolution is the SB asked angular resolution. For ACA and TP this is fixed to 10.

(c) SB Completition Score, 15%. SBs already started and closer to be completed get higher scores

(d) Letter Grade Score, 15%. Score given by Cycle and letter grade.

(e) Executive Score, 10%. Score given by the executive of the Project.

(f) Science Rank Score, 5%. Score given by the scientific ranking of the project.

2. **CODE:** Project Code
3. **SB UID:** Scheduling Block's UID
4. **SB Name:** Scheduling Block's Name
5. **SB State:** Scheduling Block's state, or status, taken from the project tracker
6. **Band:** Receiver(s) asked by the SB.
7. **RA:** Representative Right Ascension.
8. **DEC:** Representative Declination.
9. **HA:** Hour Angle for the given date and time.
10. **Elev.:** Elevation, in degrees, for the given date and time.
11. **Sets in:** Time left until the first of the field sources (science targets) goes down the horizon limit. *This calculated by checking the field sources coordinates of the SB, and not by the representative coordinates.*
12. **Grade:** Grade letter for the SB's project.
13. **Executive:** SB's project executive.
14. **Rank:** SB's Project science rank
15. **Exec. Req.:** Number of executions requested for this SB.
16. **Exec. Done.:** Number of execution blocks for this SB, that have the QA0 status set to PASS, or in Unset state.
17. **TsysFrac:** Given the TSys assumed by the PI in the OT, and the actual TSys with the given pwv, this is the multiplicative factor for the time on source (integration time) to reach the sensitivity asked by the PI. E.G., if the TSysFrac is 0.8 it means that with the 80% of the asked integration time the rms will be achieved.
18. **BLFrac.:** Given the current number of antennas and array configuration the number of usable baselines is calculated, and is compared with the SB requirements, e.g., 34 antennas for BL, 9 for ACA. The ratio of these two number gives the corrective factor needed to achieve the PI requested rms. E.G., if the factor is 1.22, it means that the ToS should be a 22% higher to achieve the rms.
19. **TotalFrac.:** The total multiplicative factor for the time on source needed given the calculated TsysFrac and BLFrac. If TotalFrac is higher than 1.3, which means that if the SB is run with these conditions the rms achieved would be sqrt(1/1.3) ~ 87% of the asked rms, the whole row will have a red background. **This does not mean the AoD should change the ToS, unless a clear policy has been given by PMG or the ES leader.**
20. **Array Score:** The array score, given for information purposes.
21. **Cond. Score:** The condition score, given for information purposes.
22. **maxPWVC:** The PWV used by the PI/P2G on the OT to calculate how much integration time is needed to get the sensitivy requested.
23. **ArrayMinAR:** The minimum array's resolution that the current SB will accept. This value comes from Stephane's script, and is corrected for all SBs to the equivalent resolution at a 100GHZ and a source that would transit at zenith.

24. **ArrCorr:** The angular resolution requiered by the SB, corrected to the equivalent resolution at 100GHz and source with DEC -23.
25. **ArrayMinAR:** The maximum array's resolution that the current SB will accept.
26. **Point Source:** are the targets of the SB point sources?
27. **TimeOnSource:** Integration time, in seconds, for the science target(s). In the case of multisources, this time should be multiplied by the number of sources.
28. **PRJ UID:** The SB's project UID.

CHAPTER

THREE

SELECTION AND SCORE ALGORITHMS

3.1 Selection and Data preparation

(wtoAlgorithm.WtoAlgorithm.selector() (page 23))

1. **Calculate observability using the pyephem libraries.**

   For all the science field sources of an SB and fixed calibration sources, we calculate the current elevation, rise LST and set LST. If a field source is a Solar System object, or an ephemeris source, we calculate first the current RA and DEC, and then the other parameters. The current elevation for the SB comes from the source with the minimun elevation; the rise LST for the SB is the LST of the source that would rise last; and the set LST for the SB is the LST of the source that would set first. The rise and set LST are calculated using the elevation limit (*horizon*) gave as an input for gWTO.

   Relavant XML child/tag or gWTO2 variables:

   - SchedBlock.FieldSource['solarSystemObject']
   - SchedBlock.FieldSource.sourceCoordinates.longitude
   - SchedBlock.FieldSource.sourceCoordinates.latitude
   - SchedBlock.FieldSource.isQuery
   - SchedBlock.FieldSource.sourceEphemeris
   - Date, horizon limit.

2. **Select SB by array type: 12m, 7m, TP.**

   Relavant XML child/tag or gWTO2 variables:

   - SchedBlock.ObsUnitControl['arrayRequested']
   - Array Type.

3. **Calculate opacity, airmass and sky Temperature.**

   For both the OT asumed conditions and current, actual, conditions, based on the current PWV.

   The implementation has several steps:

   (a) First, a table with values of Tau as fuction of PWV and representative frequencies was created. The file with this tables, in csv format, is called **tau.csv**. The frequencies are between 84.0 and 720.0 GHz, with steps of 100 MHz; pwv values are between 0.0 and 20.0 mm, in steps of 0.05. The values were calculated using the atmosphere model algorithms from CASA 4.2.1, using as input variables $P = 580.0$, $H = 20.0$, $T = 270.0$, $altitude = 5059$ and $chansep = 0.1$.

   (b) A table with Tsky values as function of PWV and representative frequencies was also create. The file with this table, in csv format, is called **tskyR.csv**. Description of columns, rows, and values is the same as the table tau.csv, the only addition, is that the Tsky in the tables assumes the airmass of a source at Zenith.

(c) Internaly, four columns are created for all SBs: *tau_org* and *tsky_org*, storing the value of tau and tsky for the conditions assumed by the OT, i.e., pwv from maxPWVC; *tau* and *tsky*, storing the values of tau and tsky for the current conditions, i.e., pwv from the gui's PWV variable.

Relavant XML child/tag or gWTO2 variables:

- SchedBlock.Preconditions.WeatherConstraints.maxPWVC
- SchedBlock.SchedulingConstraints.representativeCoordinates.latitude
- SchedBlock.SchedulingConstraints.representativeFrequency
- Date, current PWV
- Tsky and Tau tables.

4. **Calculate system Temperatures.**

   Two columns are genrated for each SB: *tsys_org* and *tsys*, one for the OT's assumed conditions and the other for the current conditions. Tsys, for both cases, is calculated using:

$$T_{sys} = \frac{1+g}{\eta_{eff} e^{-\tau \cdot \sec Z}} \left(T_{rx} + T_{sky,Z=0} \left(\frac{1 - e^{-\tau \cdot \sec Z}}{e^{-\tau}} \right) \eta_{eff} + (1 - \eta_{eff}) T_{amb} \right)$$

   Where g is the sideband gain ratio ($g = 0$ for SSB and 2SB receivers, $g = 1$ for DSB); η_{eff} is the forward efficiency, which is set to a value of 0.95; T_{rx} is the receiver characteristic temperature; τ is the oppacity for a source at zenithal distance $Z = 0$; Z is the zenithal distance of the representative source at transit: T_{sky} is the sky temperature at $Z = 0$; and T_{amb} is the ambient temperature, set to a fixed value of $270K$.

   Relavant XML child/tag or gWTO2 variables:

   - SchedBlock.SchedulingConstraints.requiredReceiverBands

5. **SBs within spectral ranges with transmission higher than 50% are first selected.**

   For gWTO1, we use to have a limit of 70%, which change to 50% when the pwv was under 0.6 mm. For gWTO2 we set the limit to 50%, and a more accurate selection is applied later using T_{sys} The transmission is calculated from the previously found tau for the current conditions:

$$e^{-\tau_{\mathrm{sb,pwv}} \cdot \sec Z_{\mathrm{sb,HA=0}}} > 0.5$$

   where $\tau_{\mathrm{sb,pwv}}$ is the oppacity for the representative source of a scheduling block *sb* and with the current PWV pwv; and $Z_{sb,\mathrm{HA=0}}$ is the zenithal distance for the representative source of *sb* at transit.

   Relavant XML child/tag or gWTO2 variables:

   - $\tau_{\mathrm{sb,pwv}}$

6. **Select SBs within given HA limits.**

$$\mathrm{minHA} < \mathrm{HA}_{\mathrm{sb,time}} < \mathrm{maxHA}$$

   (mihHA = -5 and maxHA = 3 by default).

   Relavant XML child/tag or gWTO2 variables:

   - $\mathrm{HA}_{\mathrm{sb,time}}$
   - `LST`

7. **Select SBs over the given elevation limit (20 deg. default) and that won't set for at least 1 1/2 hours.**

$$(\mathrm{elev} > \mathrm{horizon}) \ \mathrm{AND} \ (\mathrm{SB}_{\mathrm{set\ time}} > 1.5 \mathrm{hours\ from\ now})$$

   Relavant XML child/tag or gWTO2 variables:

   - `LST`
   - `Horizon Limit`

- `SB set time`

8. **Remove SBs with states Phase2Submitted, FullyObserved and Deleted.**

   Relavant XML child/tag or gWTO2 variables:

   - This information comes from the ALMA.SCHED_BLOCK_STATUS.

9. **Remove SBs that belongs to projects with status Phase2Submitted or Completed.**

   Relavant XML child/tag or gWTO2 variables:

   - This information comes from the ALMA.OBS_PROJECT_STATUS, and is crosschecked against ALMA.BMMV_OBSPROJECT

10. **Remove SBs that have names like "Do not".**

    Currently the OT is not able to handle the SB status "Deleted", so SBs that are supposed to be deleted are set to status "Suspended", and the name changed to a varation of "DO NOT OBSERVE", "Do_not_observe", "DO not observe descoped", etc., depending on the mood of the P2G. The only thing in common is the presence of a "do not". Any SB with those words in the name is removed.

    Relavant XML child/tag or gWTO2 variables:

    - SchedBlock.name

11. **Remove SBs where the number of requested executions has been achieved**

    Given the requested number of executions of a SB (executionCount), check if any EB are associated to this SB, and add up the ones with QA0 flags *Unset* and *Pass*. If this last number is equal or higher than executionCount we don't select the SB.

    When a QAO *Unset* flag is set to *Fail* or *Semipass*, the number of assoc. EB will go down, and then an SB can be back on the list. This method avoids over-observing of an SB.

    Relavant XML child/tag or gWTO2 variables:

    - SchedBlock.SchedBlockControl.executionCount
    - QA0STATUS column FROM ALMA.AQUA_EXECBLOCK

12. **Select SBs that can be executed with the current array's angular resolution**

    Using the minAR and maxAR limits, corrrected by Stéphane's script and transformed to the equivalent AR at 100GHz, we select SBs that would accept current array's configuration as set in *Array AR:*:

    $$\mathrm{minAR_{SB}} < \mathrm{AR_{array}} < \mathrm{maxAR_{SB}}$$

    Relavant XML child/tag or gWTO2 variables:

    - SchedBlock.SchedulingConstraints.minAcceptableAngResolution
    - SchedBlock.SchedulingConstraints.maxAcceptableAngResolution
    - `minAR`, `maxAR`, Stéphane's script
    - *Array AR:*

13. **Calculate tsysfrac, blfrac and frac columns**

    Three new variables are calculated for each SB, that will help to do a final selection and assign scores. `tsysfrac` is the multiplicative factor the science target integration time should be corrected by, so given the current weather conditions ($\mathrm{T_{sys}}$, τ) the execution would achieve the requested sensitivity:

    $$\mathrm{tsysfrac} = \left(\frac{\mathrm{Tsys}}{\mathrm{Tsys\_org}}\right)^2$$

    `blfrac` is the multiplicative factor the science target integration time should be corrected by to account for differences between current array's characteristics (AR, $\mathrm{Number_of_Baselines}$) and requested array:

    $$\mathrm{blfrac} = \frac{\mathrm{offered\_baselines}}{\mathrm{available\_baselines}}$$

The offered_baselines will depend on Cycle (32 for cycle 1, 34 for cycle 2).

Finally, `frac` is the total multiplicative factor accounting for both previous factors:

$$\text{frac} = \text{tsysfrac} \times \text{blfrac}$$

3.2 Score and ranking

1. **SB completion score**

   A score between 4 and 10, where 4 is given to SBs that has not been started yet, and it rises as the SB gets closer to completion.

$$\text{Score}_{\text{SB completion}} = 4 + 6 \times \frac{\text{QA0 Pass} + \text{QA0 Unset}}{\text{Expected Executions}}$$

2. **SB Grade/Cycle Score**

$$\text{Score}_{\text{grade}} = 10; \text{if grade is A} \tag{3.1}$$

$$\text{Score}_{\text{grade}} = 8; \text{if grade is B and Cycle 1} \tag{3.2}$$

$$\text{Score}_{\text{grade}} = 4; \text{if grade is B and Cycle 2} \tag{3.3}$$

$$\text{Score}_{\text{grade}} = -100; \text{if grade is C} \tag{3.4}$$

3. **SB Science Score**

$$\text{Score}_{\text{science rank}} = 10 \times \frac{\max(\text{rank}) - \text{SB}_{\text{rank}}}{\max(\text{rank})}$$

4. **SB Array Score**

$$\text{Score}_{\text{array}} = 10; \text{ if array is TP or 7m} \tag{3.5}$$

$$\text{Score}_{\text{array}} = 10; \text{ if } 0.9\text{SB}_{\text{AR}} <= \text{Array}_{\text{AR}} <= 1.1\text{SB}_{\text{AR}} \tag{3.6}$$

$$\text{Score}_{\text{array}} = 9; \text{ if } 0.7\text{SB}_{\text{AR}} <= \text{Array}_{\text{AR}} < 0.9\text{SB}_{\text{AR}} \tag{3.7}$$

$$\text{Score}_{\text{array}} = 8.5\frac{\text{Array}_{\text{AR}} - \text{SB}_{\text{minAR}}}{0.7\text{SB}_{\text{AR}} - \text{SB}_{\text{minAR}}}; \text{ if } \text{Array}_{\text{AR}} < 0.7\text{SB}_{\text{AR}} \tag{3.8}$$

$$\text{Score}_{\text{array}} = \frac{10}{1.1\text{SB}_{\text{AR}} - \text{SB}_{\text{maxAR}}} \left(\text{Array}_{\text{AR}} - \text{SB}_{\text{maxAR}}\right); \text{ if } \text{Array}_{\text{AR}} > 1.1\text{SB}_{\text{AR}} \tag{3.9}$$

5. **SB Executive Score**

$$\text{Score}_{\text{executive}} = 10; \text{default to all executives} \tag{3.10}$$

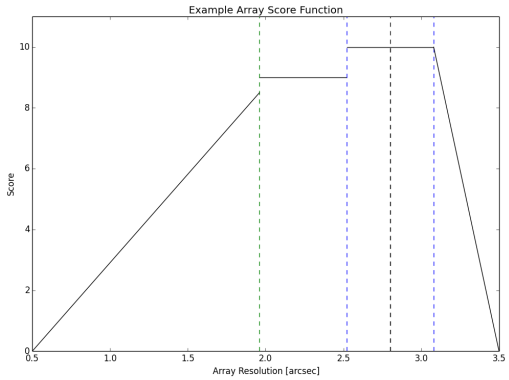

Figure 3.1: In this example, the requested array resolution ($\mathrm{SB_{AR}}$) is 2.8 arcsec, with a minimun acceptable resolution $\mathrm{SB_{minAR}} = 0.5$ and maximum $\mathrm{SB_{maxAR}} = 3.5$

6. **SB Condition Score**

$$\mathrm{Score_{cond}} = 10\left(1 - (\mathrm{frac} - 1)^2\right)\mathrm{pwv_{close}} \tag{3.11}$$

$$\mathrm{pwv_{close}} = 1 - \left|\frac{\mathrm{pwv} - \mathrm{maxPWV}}{6}\right| \tag{3.12}$$

$$\mathrm{Score_{cond}} = 10 \tag{3.13}$$

7. **SB Total Score**

3.3 Checking observability

CHAPTER

FOUR

PLAYING WITH THE LIBRARIES

Load the environment and start ipython:

```
. activateC2Test
ipython
```

Once in ipython::

```
>>> import wtoAlgorithm as wto
>>> import ephem
>>> import pandas as pd
>>> datas = wto.Algorithm(path='./wto_testing/')
```

And the run the following script. You can copy the code, and then paste into python with **%paste**, or `donwload the file`, and then load the function with **execfile('runwto.py')**:

```
    def runwto(pwv, array_name=None, d=None, num_ant=34):
        datas.query_arrays()
        if array_name == 'default':
            array_name = None
            datas.set_bl_prop(array_name)
        else:
            array_name = datas.bl_arrays.AV1.values[0]
            datas.set_bl_prop(array_name)
            datas.array_ar = 61800 / (100. * datas.ruv.max())
        if d == None:
            d = ephem.now()
        if num_ant != 34:
            datas.num_ant = num_ant
        datas.array_name = array_name
        datas.update()
        datas.date = d
        datas.pwv = pwv
        datas.selector('12m')
        datas.scorer('12m')
        print datas.score12m.sort(
            'score', ascending=False).query(
            'band != "ALMA_RB_04" and band '
            '!= "ALMA_RB_08" and isPolarization == False')[
            ['score','CODE','SB_UID','name','SB_state','band','maxPWVC', 'HA',
             'elev','etime', 'execount','Total','arrayMinAR','arcorr',
             'arrayMaxAR','tsysfrac', 'blfrac','frac','sb_array_score',
             'sb_cond_score', 'DEC','RA', 'isTimeConstrained',
             'integrationTime', 'PRJ_ARCHIVE_UID']].head(25)
        datas.num_ant_user = 34
```

The to run the wto algorithm use a pwv value between 0 and 20, with steps of 0.05 (e.g., 0.4, 0.45, but no 0.42), and assuming the latest BL Array. Set `array_name='default'` when running `runwto` (e.g. `runwto(X.XX, array_name='default')`) to use the Current configuration parameters calculated with arrayConfigurationTools and 34 antennas. Also, to change the date to current date use `runwto(X.XX, d=ephem.Date('2014-06-28 03:45')`

This will display the top 25 values of datas.scorer12m dataFrame. To check full output in an excel table run::

```
datas.score12m.to_excel('output_path/score.xls')
```

Where output_path is the full path to the directory where you want to save the score.xls excel spreadsheet.

CHAPTER

FIVE

THE WTO API

5.1 wtoDatabase.py: the gWTO database library.

This library contains the classes and functions required to query, parse and organize the Projects and SchedBlock information stored at the OSF archive in different tables.

class wtoDatabase.**ObsProject**(*xml_file*, *path='./'*)

Parameters

- **xml_file** –
- **path** –

assoc_sched_blocks()

Returns

class wtoDatabase.**WtoDatabase**(*path='/.wto/'*, *source=None*, *forcenew=False*)

WtoDatabase is the class that stores the Projects and SB information in dataframes, and it also has the methods to connect and query the OSF archive for this info.

A default instance will use the directory $HOME/.wto as a cache, and by default find the approved Cycle 2 projects and carried-over Cycle 1 projects. If a file name or list are given as 'source' parameter, only the information of the projects in that list or filename will be ingested.

Setting *forcenew* to True will force the cleaning of the cache dir, and all information will be processed again.

Parameters

- **path** (*str, default '$HOME/.wto'*) – Path for data cache.
- **source** (*list or str*) – File or list of strings with the codes of the projects to be ingested by WtoDatabase.
- **forcenew** (*boolean, default False*) – Force cache cleaning and reload from archive.

create_allsb(*split=False*, *path=None*)

Parameters

- **split** –
- **path** –

create_summary()

filter_c1()

forcenew()

get_obsproject(*code*)

Parameters **code** –

populate_newar()

populate_schedblock_info()

populate_schedblocks()

populate_sciencegoals_sbxml()

row_fieldsource(*fs*, *sbuid*, *array*, *new=False*)

Parameters

- **fs** –
- **sbuid** –
- **new** –

row_newar(*sbuid*, *new=False*)

Parameters

- **sbuid** –
- **new** –

row_schedblock_info(*sb_uid*, *new=False*)

Parameters

- **sb_uid** –
- **new** –

row_schedblocks(*sb_uid*, *partid*, *new=False*)

Parameters

- **sb_uid** –
- **partid** –
- **new** –

row_sciencegoals(*code*, *new=False*)

Parameters

- **code** –
- **new** –

Returns

row_spectralconf(*ss*, *sbuid*, *new=False*)

Parameters

- **ss** –
- **sbuid** –
- **new** –

row_target(*tg*, *sbuid*, *new=False*)

Parameters

- **tg** –
- **sbuid** –
- **new** –

start_wto()

Initializes the wtoDatabase dataframes.

The function queries the archive to look for cycle 1 and cycle 2 projects, disregarding any projects with status “Approved”, “Phase1Submitted”, “Broken”, “Canceled” or “Rejected”.

The archive tables used are ALMA.BMMV_OBSPROPOSAL, ALMA.OBS_PROJECT_STATUS, ALMA.BMMV_OBSPROJECT and ALMA.XML_OBSPROJECT_ENTITIES.

Returns None

update(*connect=True*)

Parameters **connect** –

Returns

wtoDatabase.**convert_deg**(*angle*, *unit*)

Parameters

- **angle** –
- **unit** –

Returns

wtoDatabase.**convert_ghz**(*freq*, *unit*)

Parameters

- **freq** –
- **unit** –

Returns

wtoDatabase.**convert_jy**(*flux*, *unit*)

Parameters

- **flux** –
- **unit** –

Returns

wtoDatabase.**convert_mjy**(*flux*, *unit*)

Parameters

- **flux** –
- **unit** –

Returns

wtoDatabase.**convert_sec**(*angle*, *unit*)

Parameters

- **angle** –
- **unit** –

Returns

wtoDatabase.**convert_tsec**(*time*, *unit*)

Parameters

- **time** –
- **unit** –

Returns

5.2 wtoAlgorithm.py: the gWTO selector and scorer library.

This library contains the classes and functions required to select and rank SBs from the information that is stored in a WtoDatabse object.

class wtoAlgorithm.**WtoAlgorithm**(*path='/.wto/'*, *source=None*, *forcenew=False*)

Inherits from WtoDatabase, adds the methods for selection and scoring. It also sets the default parameters for these methods: pwv=1.2, date=now, array angular resolution, transmission=0.5, minha=-5, maxha=3, etc.

Parameters

- **path** (*(default='/.wto/') String.*) – A path, relative to $HOME, where the cache is stored.
- **source** – See WtoDatabase definitions.
- **forcenew** – See WtoDatabase definitions.

Returns A WtoAlgorithm instance.

calculate_score(*ecount*, *tcount*, *srank*, *ar*, *aminar*, *amaxar*, *las*, *grade*, *repfreq*, *dec*, *execu*, *array*, *frac*, *maxpwvc*, *code*, *points*)

Please go to the *Score and ranking* (page 14) section for an algorithm's description.

Parameters

- **ecount** (*Integer.*) – Executions requested by the SB
- **tcount** (*Integer.*) – Total executions with QA0 Pass or Unset for the SB.
- **srank** (*Intenger*) – SB Science ranking.
- **ar** (*Float.*) – SB requested Angular Resolution, from the Science Goal.
- **aminar** (*Float. In arcsec.*) – The minum angular resolution the SB can accept to be observed.
- **amaxar** (*Float. In arcsec.*) – The maximum angular resolution the SB can accept to be observed.
- **las** (*Float. In arcsec.*) – SB requested Largest Angular Scale, from the Science Goal.
- **grade** (*String, can be A, B or C.*) – SB's Project letter grade.
- **repfreq** (*Float, in GHz.*) – SB's representative frequency.
- **dec** (*Float, in degrees.*) – SB's representative declination coordinates, as determined by self.check_observability().
- **execu** (*String, can be NA, EU, EA, CL or OTHER.*) – SB's Project Executive.
- **array** (*String, can be 12m, 7m or tp.*) – Array type.
- **frac** (*Float.*) – Total time fraction calculated by self.selector() for the SB to reach the required sensitivity.
- **maxpwvc** (*Float, a value between 0 and 20.*) – SB's maxPWVC variable.
- **code** (*String.*) – SB's project code.

Returns Tuple with ...

check_observability(*array*)

Parameters **array** –

Returns

query_arrays()

scorer(*array*)
Method that handles the score calculation for each array type. It applies `self.calculate_score()` to the previously selected SBs using `self.selector()`

Parameters **array** (*String.*) – '12m', '7m', 'tp'

Returns Creates a score table for the instance, which will be named as `score12m`, `score7m` or `scoretp`.

selector(*array*)
Selects SBs that can be observed given the current weather conditions, HA range, array type and array configuration (in the case of 12m array type) and SB/Project Status. See *Selection and Data preparation* (page 11)

Parameters **array** (*String.*) – '12m', '7m', 'tp'

Returns Depending on the array type, creates tables select12m, select7m or selecttp.

set_array_ar(*ar*)

Parameters **ar** –

set_arrayar(*ar*)

Parameters **ar** –

set_bl_prop(*array_name*)

Parameters **array_name** –

set_date(*date*)

Parameters **date** –

set_maxha(*ha*)

Parameters **ha** –

set_minha(*ha*)

Parameters **ha** –

set_pwv(*pwv*)

Parameters **pwv** –

set_trans(*transmission*)

Parameters **transmission** –

wtoAlgorithm.**observable**(*solarSystem*, *sourcename*, *RA*, *DEC*, *horizon*, *isQuery*, *ephemeris*, *alma*)

Parameters

- **solarSystem** –
- **sourcename** –
- **RA** –
- **DEC** –
- **horizon** –
- **isQuery** –
- **ephemeris** –
- **alma** –

Returns

wtoAlgorithm.**read_ephemeris**(*ephemeris*, *date*)

Parameters

- **ephemeris –**
- **date –**

Returns

Figure 5.1: Figure 4, UML diagram of wtoAlgorithm.py

CHAPTER

SIX

THE WTO DATA FRAMES

6.1 wtoDatabase.obsprojects

Main obsproject table ingested from queries to the archive.

COLUMN	VALUE
PRJ_ARCHIVE_UID	*(string)* Project UID (1)
DELETED	*(boolean int)* Is Deleted? (2)
PI	*(string)* Principal Investigator (3)
PRJ_NAME	*(string)* Project Name (4)
** CODE	*(string)* Project Code (5)
PRJ_TIME_OF_CREATION	*(string)* Project creation timestamp (6)
PRJ_SCIENTIFIC_RANK	*(int64)* Project Rank (7)
PRJ_VERSION	*(string)* Project Version (8)
PRJ_LETTER_GRADE	*(string)* Project Grade (9)
DOMAIN_ENTITY_STATE	*(string)* Project Status (10)
OBS_PROJECT_ID	*(string)* Project UID (11)
EXEC	*(string)* Executive (12)
timestamp	*(datetime64[ns])* Project latest update date (13)
obsproj	*(string)* Obsproject XML filename (14)

1. ALMA.BMMV_OBSPROJECT.PRJ_ARCHIVE_UID
2. ALMA.BMMV_OBSPROJECT.DELETED
3. ALMA.BMMV_OBSPROJECT.PI
4. ALMA.BMMV_OBSPROJECT.PRJ_NAME
5. ALMA.BMMV_OBSPROJECT.CODE
6. ALMA.BMMV_OBSPROJECT.PRJ_TIME_OF_CREATION
7. ALMA.BMMV_OBSPROJECT.PRJ_SCIENTIFIC_RANK
8. ALMA.BMMV_OBSPROJECT.PRJ_VERSION
9. ALMA.BMMV_OBSPROJECT.PRJ_LETTER_GRADE
10. ALMA.BMMV_OBSPROJECT.DOMAIN_ENTITY_STATE
11. ALMA.BMMV_OBSPROJECT.OBS_PROJECT_ID
12. ALMA.BMMV_OBSPROPOSAL.ASSOCIATEDEXEC
13. ALMA.XML_OBSPROJECT_ENTITIES.TIMESTAMP
14. link to ALMA.XML_OBSPROJECT_ENTITIES.XML

6.2 wtoDatabase.sciencegoals

COLUMN	VALUE
CODE	*(string)* Project Code (1)
** partId	*(string)* Science Goal partId (2)
AR	*(float64)* Desired angular resolution (arcsec) (3)
LAS	*(float64)* Largest scale (arcsec) (4)
bands	*(string)* ALMA Band (5)
isSpectralScan	*(boolean)* (6)
isTimeConstrained	*(boolean)* (7)
useACA	*(boolean)* (8)
useTP	*(boolean)* (9)
ps	*(boolean)* Is point source?
SBS	*(list of strings)* *ScienceGoals SBs* (page 26)
startRime	*(string)* Time constrain start (10)
endTime	*(string)* Time constrain end (11)
allowedMargin	*(float64)* TC allowed margin (12)
allowedUnits	*(string)* units (13)
repeats	*(int)* (14)
note	*(string)* Time constraint notes (15)
isavoid	*(boolean)* Time constrait is to avoid (16)

1. ALMA.BMMV_OBSPROJECT.CODE
2. `xml:ObsProject.ObsProgram.ScienceGoal.ObsUnitSetRef['partId']`
3. `xml:...PerformanceParameters.desiredAngularResolution`
4. `xml:...PerformanceParameters.desiredLargestScale`
5. `xml:...requiredReceiverBands`
6. `xml:...SpectralSetupParameters.SpectralScan`
7. `xml:...PerformanceParameters.isTimeConstrained`
8. `xml:...PerformanceParameters.useACA`
9. `xml:...PerformanceParameters.useTP`
10. `xml:...sciencegoal.PerformanceParameters.TemporalParameters.startTime`
11. `xml:...sciencegoal.PerformanceParameters.TemporalParameters.endTime`
12. `xml:...sciencegoal.PerformanceParameters.TemporalParameters.allowedMargin`
13. (idem)
14. `xml:...PerformanceParameters.TemporalParameters.repeats`
15. `xml:...PerformanceParameters.TemporalParameters.note`
16. `xml:...PerformanceParameters.TemporalParameters.isAvoidConstraint`

6.2.1 Getting the Scheduling Blocks of a Science Goal.

To get the Scheduling Blocks that are part of a ScienceGoal entity the partId is used.

6.3 wtoDatabase.schedblocks

COLUMN	VALUE
SB_UID	*(string)* SB UID
partId	*(string)* Science Goal partId
timestamp	*(datetime64[ns])* Project latest update date (13)
sb_xml	*(string)* SchedBlock XML filename

6.4 wtoDatabase.schedblock_info

COLUMN	VALUE
SB_UID	*(string)* SB UID
partId	*(string)* Science Goal partId
name	*(string)* SB Name
status_xml	*(string)* SB status (from xml entity)
repfreq	*(float64)* SB representative frequency (GHz)
band	*(string)* SB requested Band
array	*(string)* SB type of array
RA	*(float64)* Representative RA (degrees)
DEC	*(float64)* Representative DEC (degrees)
minAR_old	*(float64)* Original minimum angular resolution (arcsec)
maxAR_old	*(float64)* Original maximum angular resolution (arcsec)
execount	*(float64)* Requested executions
isPolarization	*(boolean)* Is a polarization SB?
amplitude	*(string)* AmplitudeParameters partId
baseband	*(string)* BasebandParameteres partId
polarization	*(string)* PolarizationParameters partId
phase	*(string)* PhaseParameters partId
delay	*(string)* DelaryParameters partId
science	*(string)* ScienceParameters partId
integrationTime	*(float64)* Science target integration time (sec)
subScandur	*(float64)* Science target subscan duration (sec)
maxPWVC	*(float64)* PWV asumed by the OT (mm)

6.5 wtoDatabase.target

COLUMN	VALUE
SB_UID	*(string)*
specRef	*(string)*
fieldRef	*(string)*
paramRef	*(string)*

6.6 wtoDatabase.fieldsource

COLUMN	VALUE
fieldRef	object
SB_UID	object
solarSystem	object
sourcename	object
name	object
RA	float64
DEC	float64
isQuery	object
intendedUse	object
qRA	object
qDEC	object
use	object
search_radius	object
rad_unit	object
ephemeris	object
pointings	float64
isMosaic	object

6.7 wtoDatabase.spectralconf

COLUMN	VALUE
specRef	object
SB_UID	object
BaseBands	float64
SPWs	float64

6.8 wtoDatabase.sb_summary

COLUMN	VALUE
CODE	object
OBS_PROJECT_ID1	object
partId	object
SB_UID	object
name	object
status_xml	object
bands	object
repfreq	float64
array	object
RA	float64
DEC	float64
minAR	float64
maxAR	float64
arrayMinAR	float64
arrayMaxAR	float64
execount	float64
PRJ_SCIENTIFIC_RANK	float64
PRJ_LETTER_GRADE	object
EXEC	object
OBSUNIT_UID	object

Continued on next page

Table 6.1 – continued from previous page

COLUMN	VALUE
NAME	object
REPR_BAND	float64
SCHEDBLOCK_CTRL_EXEC_COUNT	float64
SCHEDBLOCK_CTRL_STATE	object
MIN_ANG_RESOLUTION	float64
MAX_ANG_RESOLUTION	float64
OBSUNIT_PROJECT_UID	object
DOMAIN_ENTITY_STATE	object
OBS_PROJECT_ID	object
QA0Unset	float64
QA0Pass	float64
Total_exe	float6

6.9 wtoDatabase.qa0

COLUMN	VALUE
SCHEDBLOCKUID	object
QA0STATUS	object

6.10 wtoDatabase.scheduling_proj

Queries project data from the SCHEDULING_AOS archive tables.

```
SELECT *
FROM SCHEDULING_AOS.OBSPROJECT
WHERE regexp_like (CODE, '^201[23].*\.[AST]')
```

COLUMN	VALUE
OBSPROJECTID	SQL index
OBSPROJECT_UID	OBSPROJECT xml entity UID
CODE	Project CODE
NAME	Project Name
VERSION	Project Version
PI	Principal Investigator user name
SCIENCE_SCORE	Science Score
SCIENCE_RANK	Science Ranking
SCIENCE_GRADE	Project letter grade
STATUS	Project status in SCHEDULING_AOS
TOTAL_EXEC_TIME	
CSV	Is CSV?
MANUAL	Is Manual Mode?
OBSUNITID	Obsunit part ID
STATUS_ENTITY_ID	
STATUS_ENTITY_ID_ENCRYPTED	
STATUS_ENTITY_TYPE_NAME	
STATUS_SCHEMA_VERSION	
STATUS_DOCUMENT_VERSION	
STATUS_TIMESTAMP	

6.11 wtoDatabase.scheduling_sb

Queries scheduling block data from SCHEDULING_AOS tables.:

```
SELECT ou.OBSUNIT_UID,sb.NAME,sb.REPR_BAND,
       sb.SCHEDBLOCK_CTRL_EXEC_COUNT,sb.SCHEDBLOCK_CTRL_STATE,
       sb.MIN_ANG_RESOLUTION,sb.MAX_ANG_RESOLUTION,
       ou.OBSUNIT_PROJECT_UID
FROM SCHEDULING_AOS.SCHEDBLOCK sb, SCHEDULING_AOS.OBSUNIT ou
WHERE sb.SCHEDBLOCKID = ou.OBSUNITID AND sb.CSV = 0
```

COLUMN	VALUE
OBSUNIT_UID	object
NAME	object
REPR_BAND	int64
SCHEDBLOCK_CTRL_EXEC_COUNT	int64
SCHEDBLOCK_CTRL_STATE	object
MIN_ANG_RESOLUTION	float64
MAX_ANG_RESOLUTION	float64
OBSUNIT_PROJECT_UID	object

6.12 wtoDatabase.sbstate

CHAPTER

SEVEN

APENDIX

7.1 Assessment of Current Array's Angular Resolution

1. From the dashboard get a list of antennas and pads that are available and working in band 3 or 6. This should be saved in a file with the name YYYY-MM-DD.cfg, which is a space-separeted values file, with the antenna in the first column and pad in the second.

   (`Example file`)

2. Run the script **arrayConfiguration.py**:

   ```
   casapy -c arrayconfiguration.py -s yes -l 2h -i <path>/YYYY-MM-DD.cfg
   ```

   Where **<path>** is the path where you stored the YYYY-MM-DD.cfg file. The execution might fail complaining about missing C34-?.cfg files: copy them to the path where you are running the script from `~/AIV/science/ArrayConfiguration/Tools/Cycle2/*.cfg`

Note: **Latest update:** July 28, 2014

Latest changes in the code:

1. Fixed rank score function

Note: TODO List.

1. Deal with Total Power SBs.
2. Add all sb option, to show all SBs with a quick explanation on why they are or not observable.
3. Add explanations of different Tabs: Time Const, Polarization, Sessions, TP, etc.
4. Use time constraint info to actually show or not Time Constrained SBs in the relevant Tab.
5. Tune up the ranking algorithm. Maybe it would be a better practice to start with a base score given by what it is now called the *condition score*, and then punish or reward this score taking into account the other conditions.

Warning: Things to be checked.

1. Some SBs are Time Constrained, but they have not set the flag `isTimeConstrained`.
2. Need to handle some SBs that have multiple ScienceParameters.

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

W