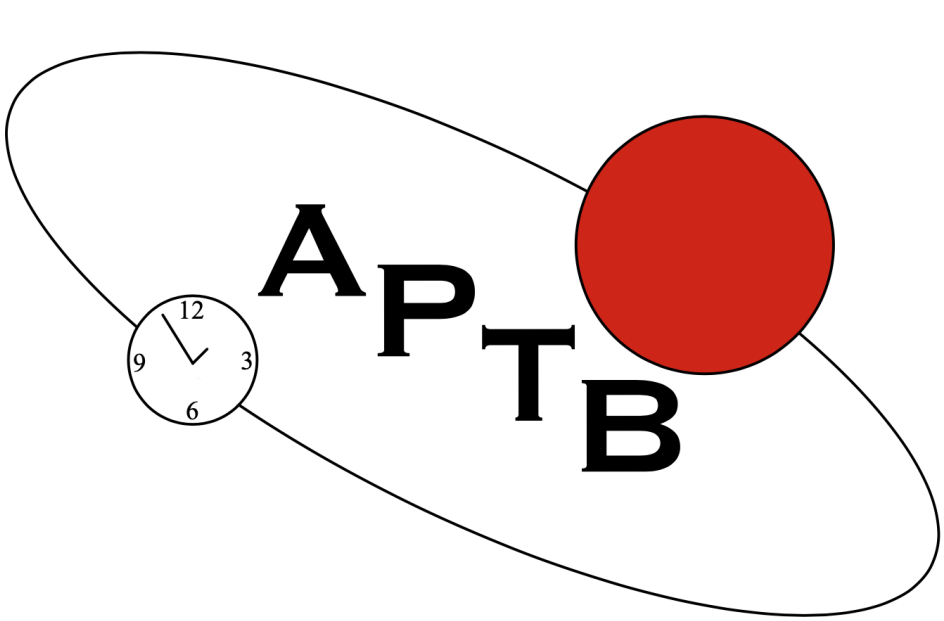



# Algorithmic Pulsar Timer for Binaries User Guide

Version 0.9

Jackson Taylor

August 16, 2023

# Contents

# 1 Introduction

Pulsar timing can be a tricky and time-consuming process, especially in the case of binary pulsars due to their higher-dimensional parameter space. In effect, it takes longer to time to binary pulsars over isolated pulsars. Binary pulsars with sparse or low-quality times of arrival (TOAs) can take hours, days, or longer to solve. This is why a fully-automated algorithm can be a great time saver to phase-connect easier pulsars and a necessity to time trickier pulsars. A human would heavily object to the idea of spending tens of hours timing a pulsar with no guarantees of success while an algorithm would carry on without a fuss, and potentially prevail in less than half the time.

The Algorithmic Pulsar Timer for Binaries (APTB[1]) is software that uses the Python coding language and the PINT[2] pulsar timing Python package to phase connect newly discovered binary pulsars. This guide is intended for those who simply want to use APTB to phase connect pulsars rather than understand how APTB works. If you do want to understand APTB's methods, I point to Taylor & Ransom (2023, in preparation)[3]. While I will not assume you have read the APTB paper, I will assume you know the basics of pulsar phase connection and timing. If you successfully time a pulsar using APTB, I kindly ask that you cite the APTB paper. If you have any questions or suggestions, please feel free to contact me at jacksondtaylor4@gmail.com.

**User Guide Goals**: This user guide hopes to aid APTB users with

1. how to get started,
2. how to navigate the plethora of files APTB creates,
3. useful tips & tricks to crack tough pulsars,
4. and suggestions as to why APTB may fail on some occasions.

# 2 APTB's Interface Via the Command Line

The user can interact with APTB via the command line allowing them to fine-tune how APTB times a pulsar. This is fairly straightforward for anyone accustomed to the command line. For most arguments, the default values are usually optimal and so I recommend only changing a few, if any. The only

[1] https://github.com/Camryn-Phillips/APT
[2] https://github.com/nanograv/PINT
[3] [put the full citation here]

required arguments are the `par` and `tim` file names which are also the only positional arguments. Every other argument is either an option that takes a value or an on/off flag. If you would like to see any additional command line features, feel free to contact me, or you can also fork the GitHub repository and edit the code yourself. It should be noted that APTB has been extensively tested on Mac OS and Linux machines. APTB uses code that should also work on Windows, but this has not been verified.

## 2.1 Getting Started

To time any pulsar, you need a sufficient number of TOAs that span at least a year (and hopefully longer than 5 years). This is in the form of a `tim` file. You also need the associated `par` file needed to obtain these TOAs. This `par` file should be a good initial model, and APTB will hopefully turn it into a great model. The fastest out-of-the-box use of APTB is as follows. Simply put the `par` and `tim` files into the same directory as APTB , and run the following in your terminal

```
python APTB.py [parfile] [timfile]
```

This runs APTB on its default argument values. If you do not have the necessary Python packages, like NUMPY, PINT, and TREELIB for example, this is when you will find out. Install the needed packages into your relevant Python environment until the above command starts to run without any immediate errors. I refer to Section 4 for any other errors that may occur at this stage.

The options that take a value can be used as follows. In general, for a particular option name and its value, run

```
python APTB.py [parfile] [timfile] --[option name] [value]
```

For example, if you want APTB to only start fitting for right ascension after the phase connected unJUMPed TOAs span 50 days, you would input

```
python APTB.py [parfile] [timfile] --RAJ_lim 50
```

where `RAJ_lim` is the option name and 50 is its value.

The on/off flags are used in a similar way. If you want a flag to be on:

```
python APTB.py [parfile] [timfile] --[flag_name]
```

If you want a flag to be off:

```
python APTB.py [parfile] [timfile] --no-[flag_name]
```

For example, if you want there to be no start warning:

```
python APTB.py [parfile] [timfile] --no-start_warning
```

where `start_warning` is the flag name.

## 2.2 Most Used Command Line Features

Most command-line arguments are fine to be kept at their default values. That being said, it is sometimes necessary to change some arguments to successfully phase connect a pulsar. The most used arguments will be explained in this section. See Section 2.1 on how to use the different types of command line arguments.

### 2.2.1 Most Used Options that Take a Value

`starting_points` This specifies the first observation that should be un-JUMPed. APTB will begin phase connection from this observation. The default is the 5 densest observations as calculated in Section 2.1 of Taylor and Ransom (2023, in preparation), which is repeated in Equation 1 in Section 2.3.1 of this paper. Each observation is designated by its order chronologically, starting from 0. For example, the 3rd observation means the `start_points` value would be 2. Multiple starting points can be specified by adding a comma between each integer with no spaces in between. APTB will attempt all starting points.

`RAJ_lim` - Default 1.5 (days). This will specify the phase-connected TOA span in days that right ascension *can* be fit for. Once the TOAs that are phase connected span more than this limit, APTB will fit for right ascension only if the model would significantly improve. Unless otherwise stated, all other `_lim` argument types operate in the same way.

`DECJ_lim` - Default 2.0 (days). This will specify the phase-connected TOA span in days that declination can be fit for.

`F1_lim` - This will specify the phase-connected TOA span in days that the pulse frequency time derivative ($F1$) can be fit for. The default is the span where a lack of $F1$ would cause a residual of 0.35 phase based on the prediction of $F1$ from the standard $P - \dot{P}$ diagram.

`EPS_lim` - Default $5 \cdot PB$, where $PB$ is the binary orbital period. This will specify the phase-connected TOA span in days that both $\epsilon_1$ and $\epsilon_2$ can be fit for. Only relevant for the ELL1 binary model.

`ECC_lim` - Default 0.0 (days). This will specify the phase-connected TOA span in days that the eccentricity ($e$) can be fit for. Only relevant for the BT binary model. By default, $e$ will be fit for immediately.

`OM_lim` - Default 0.0 (days). This will specify the phase-connected TOA span in days that the longitude of periastron ($w$) can be fit for. Only relevant for the BT binary model. By default, $w$ will be fit for immediately.

### 2.2.2 Most Used On/Off Flags

`start_warning` - Default on. If the reduced $\chi^2$ ($\chi^2_\nu$) is $\chi^2_\nu > 3$ at the very beginning where every observation day is properly JUMPed, APTB will stop. If off, APTB will not stop here. In other words, if $\chi^2_\nu > 3$ at the beginning and you would like APTB to attempt phase connection anyway, you should turn this off.

## 2.3 Every Command Line Feature Explained

This section will explain every command line feature offered. Usually, a vast majority of these are kept at their defaults. I try to explain everything without going too much into detail.

### 2.3.1 Every Option that Takes a Value

`starting_points` This specifies the first observation that should be un-JUMPed. APTB will begin phase connection from this observation. The default is the 5 densest observations as calculated in Section 2.1 of Taylor and Ransom (2023, in preparation). Each observation is designated by its order chronologically, starting from 0. For example, the 3rd observation means the `start_points` value would be 2. Multiple starting points can be specified by adding a comma between each integer with no spaces in between. APTB will attempt all starting points.

`RAJ_lim` - Default 1.5 (days). This will specify the phase-connected TOA span in days that right ascension *can* be fit for. Once the TOAs that are phase connected span more than this limit, APTB will fit for right ascension only if the model would significantly improve. Unless otherwise stated, all other `_lim` argument types operate in the same way.

`DECJ_lim` - Default 2.0 (days). This will specify the phase-connected TOA span in days that declination can be fit for.

`F0_lim` - Default 0.0 (days). This will specify the phase-connected TOA span in days that pulse frequency ($F0$) can be fit for. By default, $F0$ will be fit for immediately.

`F1_lim` - This will specify the phase-connected TOA span in days that the pulse frequency time derivative ($F1$) can be fit for. The default is the span where a lack of $F1$ would cause a residual of 0.35 phase based on the prediction of $F1$ from the standard $P - \dot{P}$ diagram.

`F1_sign_always` - Options are +, −, or None. This specifies the required sign of $F1$ for the *entire* run time. If APTB encounters a model where the sign of $F1$ differs from the sign specified here, it will prune the model. If `F1_sign_always` is the default, this type of pruning will not occur.

`F1_sign_solution` - Default $-$. Options are +, $-$, or None. This specifies the required sign of *F*1 for only when APTB thinks it found a solution. If the sign of *F*1 differs from this sign, the solution is discarded. If `F1_sign_solution` is None, no potential solutions will be discarded based on the sign of *F*1.

`F2_lim` - Default $\infty$, i.e. by default the second pulse frequency time derivative (*F*2) will never be fit for. This will specify the phase-connected TOA span in days that *F*2 can be fit for.

`EPS_lim` - Default 5 $\cdot$ *PB*, where *PB* is the binary orbital period. This will specify the phase-connected TOA span in days that both $\epsilon_1$ and $\epsilon_2$ can be fit for. Only relevant for the ELL1 binary model.

`ECC_lim` - Default 0.0 (days). This will specify the phase-connected TOA span in days that the eccentricity (*e*) can be fit for. Only relevant for the BT binary model. By default, *e* will be fit for immediately.

`OM_lim` - Default 0.0 (days). This will specify the phase-connected TOA span in days that the longitude of periastron ($\omega$) can be fit for. Only relevant for the BT binary model. By default, $\omega$ will be fit for immediately.

`Ftest_lim` - Default 0.0005. The statistical F-test is used to determine if fitting for an additional parameter would significantly improve a model. This is done by calculating the probability that the additional parameter improved the model only by chance. If this probability is lower than the `Ftest_lim`, than the parameter will be added provided that the parameter's span limit has been exceeded. Lowering `Ftest_lim` will require parameters to more significantly improve the model, while raising `Ftest_lim` will slightly loosen this requirement.

`prune_condition` - When the best phase wrap still yields a $\chi^2_\nu$ less than `prune_condition`, it will be pruned. If `branches` is on, other models will be investigated next. When every observation except one has JUMPs applied and every TOA is fit for, the $\chi^2_\nu$ here is known as the base $\chi^2_\nu$, or $\chi^2_{\nu,\,\mathrm{base}}$. The default `prune_condition` is $\chi^2_{\nu,\,\mathrm{base}}$ + 1.

`vertex_wrap` - Default 1. APTB uses the parabolic dependence of the $\chi^2_\nu$ on the phase wraps to calculate the best phase wrap at each step. To make sure the calculated vertex of this parabola is indeed the minimum, APTB will check phase wraps `vertex_wrap` above and `vertex_wrap` below this value. For example, if `vertex_wrap` is 1 and the calculated vertex is at a phase wrap of 6, APTB will check the phase wraps of 5 and 7 as well.

`data_path` - Default [present working directory]. This specifies the absolute file path where the files from the run will be stored. By default, this will be the directory in which you ran APTB.

`parfile_compare` - Default off, but `parfile_compare` is not an on/off flag. A seperate par file can be the value here. This par file will be compared with potential solutions to see if they identify the same pulse numbers. This

is for ease of use if you intend on running APTB as a way of comparing your own solution to the algorithm's solution.

`chisq_cutoff` - Default 10. When a potential solution is found, if its $\chi^2_\nu$ is above `chisq_cutoff`, then it is discarded.

`max_starts` - Default 5. APTB will look at the best `max_starts` starting points based on the TOA score as calculated in Equation 1 below.

`maxiter_while` - Default 1. When PINT fits TOAs to a model, it does so until the $\chi^2_\nu$ does not change significantly from fit to fit or until it reaches a maximum iteration number. `maxiter_while` specifies this maximum iteration number when inside the main loop of the algorithm. Increasing this number may help models settle, though usually this effect is insignificant. A higher `maxiter_while` also increases the overall run time.

`score_power` - Default 0.3. To determine which observation is not JUMPed at the beginning, a score is assigned to every TOA. The observation with the highest-scored TOA is selected at this unJUMPed observation. Phase connection then spreads out from this observation. The $i^{\text{th}}$ TOA, $t_i$, has a score,$n_i$, of

$$n_i = \sum_{j \neq i}^{N_{TOAs}} \frac{1}{|t_i - t_j|^\alpha}, \tag{1}$$

where $\alpha$ is the `score_power`. Raising $\alpha$ more strongly emphasizes dense observations. Lowering $\alpha$ emphasizes several observations that are broadly close to each other.

`min_log_level` Default INFO. Options are TRACE, DEBUG, INFO, WARNING, and ERROR. APTB updates you on its progress via the terminal with logs from LOGURU and `print` statements. `min_log_level` specifies the minimum log level that APTB will put into the terminal. For more details, see `pint.logging.setup`.

`depth_pursue` - Default $\infty$, i.e. this setting is off by default. Past this depth, APTB will not prune the model regardless of $\chi^2_\nu$. Depth here is defined as the number of observations that have been phase connected minus 1.

### 2.3.2 Every On/Off Flag

`start_warning` - Default on. If $\chi^2_\nu > 3$ at the very beginning where every observation day is properly JUMPed, APTB will stop. If off, APTB will not stop here. In other words, if $\chi^2_\nu > 3$ at the beginning and you would like APTB to attempt phase connection anyway, you should turn this off.

`final_fit_everything` - Default on. After APTB finds a potential solution, it will fit for all parameters if `final_fit_everything` is on. This may be undesirable if some parameters are known before attempting phase connection. For example, if a X-ray satellite has a precise measurement of

sky position, you would likely not want to fit for sky position on a potential solution.

`plot_final` - Default off. Whether to show the plot of a potential solution when that solution is found. The plot will be saved as a png regardless so this only affects if it immediately shows on the screen. If on, APTB will pause until the plot is closed, so this option is recommended to be kept off unless you are confident APTB will find only one solution.

`check_phase_wraps` - Default on. This determines if APTB will check for phase wraps. Turning this off will greatly lower the run time, but the chance of success is small. If you do succeed without checking for phase wraps, an algorithm like APTB was likely unnecessary.

`branches` - Default on. While APTB checks for the best phase wrap, its real strength comes in its ability to check all phase wraps with $\chi^2_\nu$ less than the `prune_condition`. APTB will do this if `branches` is kept on. The phase wraps with $\chi^2_\nu$ less than the `prune_condition` will eventually be explored in their own right after the phase-wrap paths before them have been pruned.

`try_all_masks` - Default off. Whether to loop through different starting observations after having found at least one potential solution.

`find_all_solutions` - Default on. Whether to search for more potential solutions after finding the first potential solution. This is distinct from `try_all_masks`.

`save_state` - Default on. Whether to save intermediate models into a par file. Turning this off will decrease both the run time and the number of files created. However, turning off `save_state` makes diagnosing problems or retrieving potential solutions that APTB pruned anyway nearly impossible.

`pre_save_state` - Default off. Before APTB checks for phase wraps, it will save the model if `pre_save_state` is on. `pre_save_state` being on could help with troubleshooting, but the models from the ordinary `save_state` setting are usually sufficient.

`debug_mode` - Default on. Many of the terminal messages can be muted by turning this off.

## 2.4 Way to Run APTB More Conveniently (Optional)

For those who will use APTB more than once, I strongly suggest you add the APT directory into your `PYTHONPATH`. This can be done by putting the following into your command line or `.bashrc` file.

```
export PYTHONPATH="[absolute path to APT directory]:$PYTHONPATH"
```

Then you need to add a symbolic link of APTB into your home `bin` directory.

```
ln -s [absolute path to APT directory]/APTB.py $HOME/bin/APTB
```

Make sure `bin` is in your `PATH`.

```
PATH="$HOME/bin:$PATH"
```

If everything is set up correctly, then APTB can be in *any* directory, not just the directory with the relevant `par` and `tim` files. This allows you to simply navigate to the directory with the relevant `par` and `tim` files and treat APTB like a terminal command.

```
APTB [parfile] [timfile]
```

This may seem like a small improvement in use but can help significantly with organization and workflow if you intend on using APTB more than a few times.

# 3 The Files Explained

APTB can produce hundreds of files that allow you to analyze its progress. In the scenarios where APTB found no solutions but was close, these files are crucial. At each iteration, par and tim files are created, as well as a png of the residuals plotted against TOA mjd. In Section 3.1 I discuss on a high level what APTB is doing and the solution tree it produces. In Section 3.2 I discuss how to use this tree to find the intermediate model you are looking for.

## 3.1 Solution Tree

In the process of phase connection, JUMPs are successively removed. When a JUMP is removed from an observation, APTB evaluates if the model-assigned pulse numbers are correct and applies a phase wrap to potentially correct the pulse numbers. Because the model is at least somewhat accurate, the phase wrap should be of order unity, if not zero. All phase wraps that yield a $\chi^2_\nu$ below the prune condition (see Section 2.3.1) are accepted. The phase wrap with the lowest $\chi^2_\nu$ will be explored first, with the others being explored once the temporarily-best model is pruned. The temporarily-best model will try to unJUMP another observation which, in turn, comes with its own set of acceptable phase wraps. This sets up a depth-first search solution tree. See Figure 1.

APTB saves the structure of the solution tree along with three pieces of information of each model.

1. The global iteration of the model is saved. This is the number of times APTB has started its main loop and thus corresponds to how many models were explored before this model.

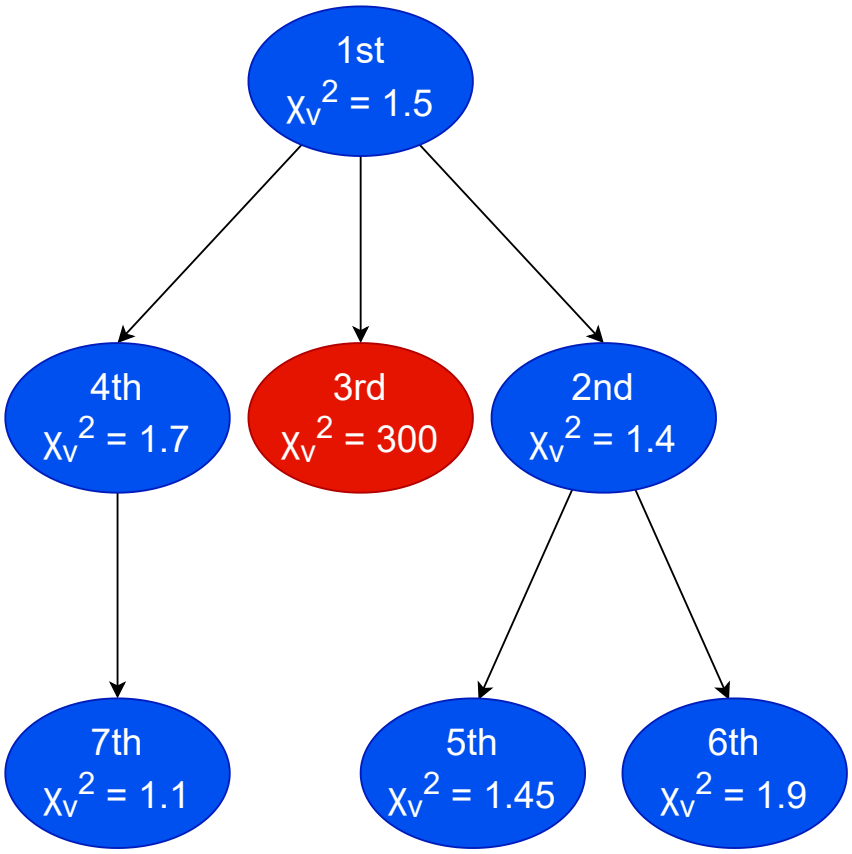



Figure 1: Flowchart of APTB's branch searching priority. Even though the bottom left model has the lowest $\chi^2_\nu$, its parent prevented it from being immediately explored. Keep in mind no path is visible until it is attempted, so the 3rd and 4th models had to be attempted to know the 2nd model was the best. The red model was pruned so none of its children are explored.

2. The depth of the model is saved. This marks how many observations had been phase connected for the current model.

3. The phase wrap decision is saved. This denotes what phase wrap APTB applied on the observation being unJUMPed.

## 3.2 How to Navigate the Files From Every Iteration

Go to the directory where the files are being saved. Then change directory to `alg_saves/[pulsarname]`. There should be files named `mask#_cluster#`, where the first number denotes the order of the starting points and the second number is the value of the starting point. For example,

`mask2_cluster7` means this was the third starting point that APTB looked at and that starting point was the 8th observation. Navigate into any of these files, with preference towards the `mask0` file as this was the highest ranked observation (see Equation 1).

If there a `Solutions` directory present, go into it. Otherwise, see the next paragraph. The `Solutions` directory holds all potential solutions from the relevant starting cluster. If there is only one such potential solution, the odds are that it is the correct one. This is especially true if the maximum depth (i.e. the number of observations) $\gtrsim 15$. If there is more than one potential solution, there is likely several ($> 10$) more. If there are tens of potential solutions, it is likely that none of them are correct. This is likely caused but too few observations as the models cannot constrain the limited data enough to exclude all but one solution. In my experience, when there are too many potential solutions they often have too high $F1$ for their $F0$.

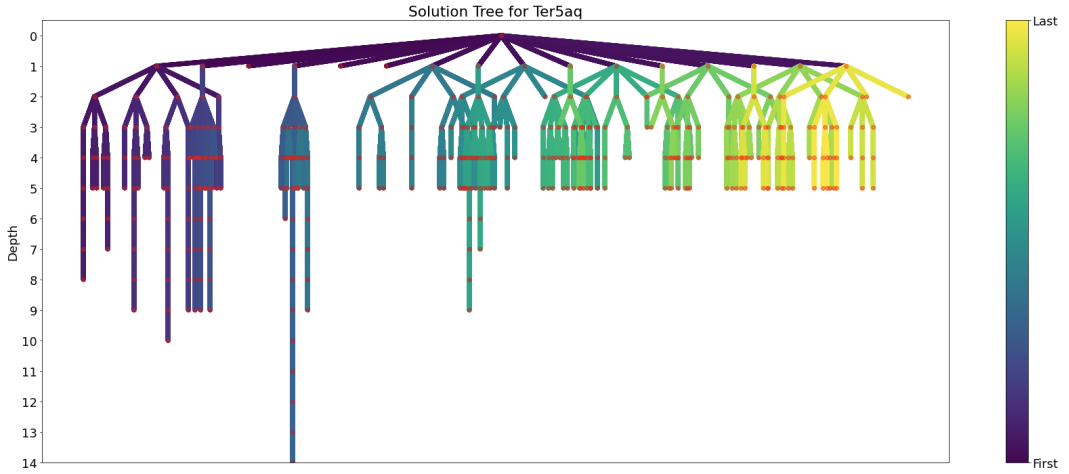
A good way to assess the different models and phase wraps that APTB explored is to find and open the file named `explore_solution_tree.tree`. This file shows a schematic of the solution tree's structure. For each node, the *i* denotes the global iteration, the *d* denotes the depth, and the *w* denotes the phase wrap decision (see Section 3.1). The files in `mask#_cluster#` should be labeled the same way. It is possible that instead of *i#*, a node has a *iU#*. This occurs if APTB stopped running due to an error (or keyboard interruption) and therefore the iteration number is unknown. The number in *iU#* has no meaning other than to uniquely identify every node. If the number of observations is high, then it's possible APTB was very close to finding the solution but failed due to not including parameters that only have effects on long timescales, like $F2$, *PBdot*, or $A1$*dot*, for example. In this case, APTB likely got to a high depth for one path, but relatively low depths for other paths. Such an example is shown in Figure 2. These instances are easy to spot in the solution tree file. They are marked by plenty of depth $< 10$ nodes, but only one path exceeding a depth of $\sim 15$. Find the node with the highest depth and note its name. Find the matching par and tim files. If the depth is high enough, you should be able to remove all JUMPs and phase connect the rest of the TOAs manually by eye.

# 4 Trouble Shooting/Common Issues

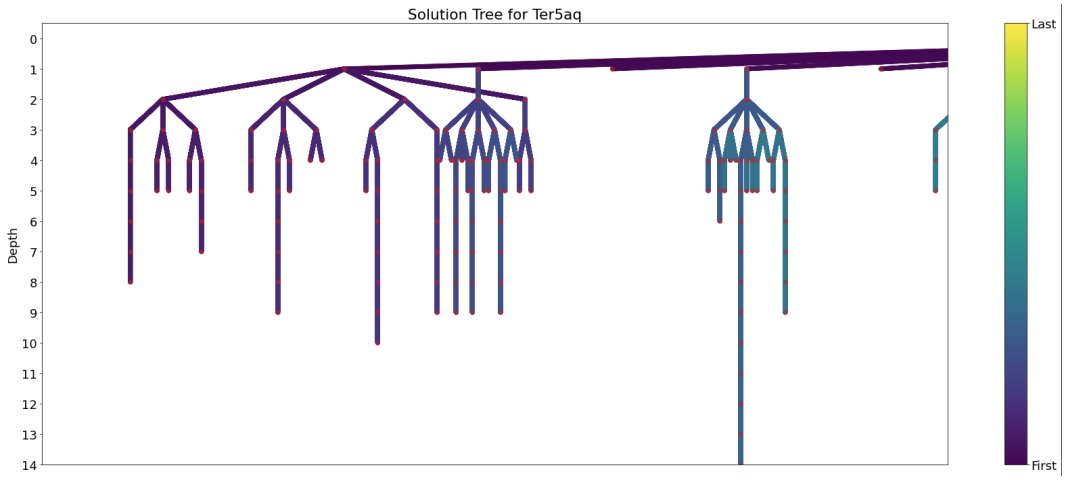
If all packages are installed, the first error you could come across would be a recursion error which would read

```
RecursionError: In start: maximum recursion depth exceeded (3)
```

This means the reduced $\chi^2$ ($\chi^2_\nu$) is greater than 3. This indicates either that




(a) A broad view of the total tree. The dark blue-turquoise path that goes past a depth of 14 actually ends at a (not shown) depth of 30.




(b) The solution tree but zoomed in on the leftmost tree structure. The overall correct dark blue-turquoise path is also shown here. Clearly, several phase wraps were investigated before APTB found and *stayed on* the correct solution path.

Figure 2: A schematic of the solution tree APTB investigated in order to time PSR 1748-24aq (Ter5aq). The path order that APTB took is qualitatively described by the color bar. The dark purple lines show paths that APTB investigated first with the light green and yellow lines showing paths APTB got to last. The depth of the solution tree is equivalent to the number of clusters that had been unJUMPed at that stage. The green and yellow paths were investigated heavily, but no meaningful solution was found. This emphasizes that the depth-30 path is indeed the unique solution.

the initial model from the pulsar's discovery is insufficient for long-term phase connection or that the error bars are estimated much lower than they should be. If you want to attempt phase connection anyway, simply add `--no-start_warning` to the command line.

Continuing the discussion from Section 3.2, there can be two main ways APTB can be unsuccessful. One being if APTB finds too many solutions. This means the `Solutions` directory is present but filled with tens of potential solutions. This is usually the case when the number of observations is about ten or less. The lack of observations admits a larger acceptable parameter space which in turn allows different sets of TOA pulse numbers that all give $\chi^2_\nu \sim 1$ (or $\chi^2_\nu \sim \chi^2_{\nu,\,\text{base}}$ if $\chi^2_{\nu,\,\text{base}}$ is not close to 1). All of these solutions are likely wrong, and they often give odd parameter values.

The other main way APTB can fail is if there are no solutions at all. Checking the `explore_solution_tree.tree` file can remedy this, provided there is a path reaching a depth much higher than other paths (Section 3.2). If no such path exists, this means APTB is being too exclusionary in its pruning conditions. It is pruning or not even finding the correct path. Some quick fixes to this include increasing `prune_condition`, increasing `max_starts` (or adding more `starting_points`), or switching the binary model. Alternatively, some limits could be relaxed by allowing $\epsilon_1$ & $\epsilon_2$ (or eccentricity & $\omega$) to be fit or lowering the minimum fit time span on sky position or *F*1. In general, APTB not finding any potential solution is a much harder problem to fix than APTB finding too many potential solutions.

# 5 Examples

I will now discuss how I would time some make-believe pulsars.

## 5.1 Example 1

Let's say I have TOAs on PSR 1. My par file is named `psr1.par` and tim file is named `psr1.tim`. I have Chandra X-ray data on this pulsar and so have a good idea of the sky position. Therefore I do not want to fit for the sky position. My TOAs span one year. I would therefore run

```
python APTB.py psr1.par psr1.tim --RAJ_lim 5000 --DECJ_lim 5000
```

5000 here means APTB will only fit for right ascension and declination after the phase connected TOAs span 5000 days. Therefore sky position will only be fit for at the very end since the TOAs only span $\sim$ 360 days. If I wanted to prevent fitting for sky position at the end I would turn off `final_fit_everything`

```
python APTB.py psr1.par psr1.tim --RAJ_lim 5000 --DECJ_lim 5000
--no-final_fit_everything
```

If after running I get an error reading

```
RecursionError: In start: maximum recursion depth exceeded (3)
```

and I want to ignore this, I would turn off the `start_warning`

```
python APTB.py psr1.par psr1.tim --RAJ_lim 5000 --DECJ_lim 5000
--no-final_fit_everything --no-start_warning
```

## 5.2 Example 2

I just timed PSR 2. My par and tim files are `psr2.par` and `psr2.tim`, respectively. My TOAs span 4 years. I know my pulsar's orbit is incredibly circular so not only will I use the ELL1 model, but I will also prevent APTB fitting for $\epsilon_1$ and $\epsilon_2$ by running

```
python APTB.py psr2.par psr2.tim --EPS_lim 5000
```

It is `EPS_lim` for both $\epsilon_1$ and $\epsilon_2$, *not* `EPS1_lim` and `EPS2_lim` individually. If I am worried that APTB will waste time looking at models with positive $F1$, I would specify $F1$'s sign should be negative or 0 throughout the entire run time, not just the end.

```
python APTB.py psr2.par psr2.tim --EPS_lim 5000 --F1_sign_always -
```

## 5.3 Example 3

I just timed PSR 3. My par and tim files are `psr3.par` and `psr3.tim`, respectively. My TOAs span 10 years. The 10th, 11th, 12th, 13th, and 14th observations are very close to each other. If I want APTB to start phase connection from the 13th observation, and APTB is not already doing this by default, I would run

```
python APTB.py psr3.par psr3.tim --starting_points 12
```

The value of `starting_points` is 12 because the counting starts from 0, not 1. Since the TOA span is relatively high, parameters that APTB will not include by default, like $F2$, will become important. Therefore, APTB will likely not fully find a solution, but it will likely pursue the correct path to a high depth. I can find the relevant high-depth model in `alg_saves/psr3/mask0_cluster12` with the help of the `alg_saves/psr3/mask0_cluster12/explore_solution_tree.tree` file (See Section 3) to finish the rest of the phase-connection process manually.