Algorithmic Pulsar Timer for Binaries

Jackson Taylor,[1, 2] Scott Ransom,[3] and Prajwal V. Padmanabh[4, 5]

[1]*Department of Physics and Astronomy, West Virginia University, P.O. Box 6315, Morgantown, WV 26506, USA*
[2]*Center for Gravitational Waves and Cosmology, West Virginia University, Chestnut Ridge Research Building, Morgantown, WV 26505, USA*
[3]*National Radio Astronomy Observatory, 520 Edgemont Rd., Charlottesville, VA 22903, USA*
[4]*Max Planck Institute for Gravitational Physics (Albert Einstein Institute), D-30167 Hannover, Germany*
[5]*Leibniz Universität Hannover, D-30167 Hannover, Germany*

ABSTRACT

Pulsar timing is a powerful tool that, by accounting for every rotation of a pulsar, precisely measures the spin frequency, spin frequency derivatives, astrometric position, binary parameters when applicable, properties of the ISM, and potentially general relativistic effects. Typically, this process demands fairly stringent scheduling requirements for monitoring observations as well as deep domain knowledge to "phase connect" the timing data. We present an algorithm that automates the pulsar timing process for binary pulsars, whose timing solutions have an additional level of complexity, although the algorithm works for isolated pulsars as well. Using the statistical F-test and the quadratic dependence of the reduced χ^2 near a minimum, the global rotation count of a pulsar can be determined efficiently and systematically. We have used our algorithm to establish timing solutions for two newly discovered binary pulsars, PSRs J1748−2446aq and J1748−2446at, in the globular cluster Terzan 5, using ∼ 70 Green Bank Telescope observations from the last 13 years.

Keywords: Binary pulsars — Pulsars — Millisecond pulsars — Radio pulsars — Pulsar timing method — Algorithms

1. INTRODUCTION

Pulsar timing is the process of monitoring and tracking pulses and times of arrival (TOAs). Requiring that each TOA unambiguously corresponds to an integer rotation of the pulsar has made the study of pulsars remarkably precise and quite fruitful since their discovery in 1967 (Hewish et al. 1968). Binary millisecond pulsars (MSPs) in particular have allowed pulsar astronomers to validate general relativity (Einstein 1915) at the 1.3×10^{-4} level in the strong field regime (Kramer et al. 2021). Timing these binary pulsars can also constrain the mass of the neutron star, ruling out several equations of state (e.g. Demorest et al. 2010; Antoniadis et al. 2013; Cromartie et al. 2020). The International Pulsar Timing Array (IPTA; Hobbs et al. 2010) also places heavy importance on timing MSPs. The IPTA is a consortium of consortia with the intention of detecting nHz-frequency gravitational waves (GWs) by correlating the data from around 100 MSPs, most of which are in binary systems. Recent evidence for the detection of these GWs (e.g. Agazie et al. 2023; Antoniadis et al. 2023; Reardon et al. 2023) has solidified the value in timing MSPs for years to come.

Early on in the pulsar timing process, the global rotation count, or pulse count, from the available data must be determined. When every TOA is assigned the correct pulse count,

this is referred to as the *solution*. In some cases, the data is too sparse to obtain a solution manually. With the increasing demand for telescope time, it becomes imperative to develop techniques for achieving a phase-connected solution with a minimal amount of data. Thus, the Algorithmic Pulsar Timer (APT) was developed by Phillips & Ransom (2022, hereafter PR22) to solve single system pulsars. However, APT is only successful if the best available model incorrectly predicts the pulse numbers by no more than roughly one or two rotations between each pair of observations. Additionally, this timer falls short of a necessary tool for solving binary systems, that is, JUMPs. A JUMP is a statement applied to a group of several TOAs that allows the fitting software to fit for an arbitrary time offset for that group. Another pulsar timing algorithm, developed by Freire & Ridolfi (2018, hereafter FR18) called DRACULA,[1] has seen success. Their algorithm uses JUMPs and works on binaries, but was not explicitly designed for binaries. The fact that DRACULA is not fully automated can indeed present a problem for phase-connecting binary pulsars more so than for isolated pulsars due to the need to account

---

[1] https://github.com/pfreire163/Dracula

for binary parameters. `DRACULA` also does not directly incorporate a procedure for when to fit for additional parameters.

We have developed the Algorithmic Pulsar Timer for Binaries (APTB)[2] to address the shortcomings of APT and `DRACULA`. As APTB was explicitly designed for binaries, it can use any of three different and well-established binary models automatically, and additional binary models can easily be implemented. Our algorithm uses the F-test for model comparison so that a parameter is only fit for when its omission becomes significant. For example, this avoids fitting for right ascension before it is no longer covariant with the spin frequency derivatives. We also implemented a straightforward tree structure to not only allow APTB to pursue several options (Section 3), but also to easily identify how exactly APTB phase-connected or failed to phase-connect the pulsar. Lastly, APTB is written in Python which permits ease of readability and has the goal of being used by a wider audience. As such, we created a user guide for those who only need to use APTB for pulsar timing rather than an explanation of our methods. It should be noted that the infrastructure required to time binary pulsars is more than sufficient to allow APTB to time isolated pulsars, as well. By combining elements of the original APT and `DRACULA`, as well as our own additions, APTB explores the best pulse counts in order to potentially yield a phase-connected fit. Despite the value of timing binary pulsars, the difficult process is not sufficiently documented in the literature. Thus, an important aspect of this paper is to document one such binary timing method.

2. TECHNIQUES AND METHODS

Residuals are the difference in phase between the model-predicted TOAs and the observed TOAs. Residuals are modulo the pulse period because initially, we cannot claim to know the global rotation count of every TOA. It is usually immediately clear when an initial timing ephemeris does not correctly assign the correct global rotation counts because the (incorrect) solution is characterized by seemingly random residuals versus pulse phase (e.g. Figure 1). Moreover, the initial timing ephemeris holds little predictive power. However, the initial model should be able to ensure that the relative rotation count within a day is correct, i.e. the TOAs in an observation are *phase connected*. TOAs in a single observation belong to a *cluster*. Given the high-dimensional parameter space involved with timing binaries, and multiple important timescales (e.g. orbital period for binary parameters, annual modulations from astrometric position fitting, and longer timescales, potentially, for spin-down effects), it is necessary to use the information from every TOA, especially those that are known to be phase-connected to other nearby
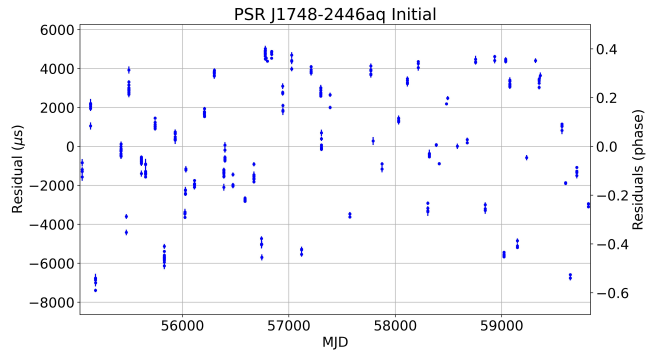

Figure 1. Residuals of the initial timing ephemeris of PSR J1748−2446aq. Notice how the residuals seem to lack any long-term trend and vary greatly from zero. This is in contrast to Figure 4 where the residuals are nearly zero with respect to their estimated error.

TOAs, even if TOAs between observations are not phase connected. By using JUMPs (Section 2.1), the residuals within an observation can be minimized with respect to the model parameters.

This improves the model, especially the binary parameters. The process of removing the JUMPs successively is known as *mapping a gap* and is described in Section 2.4 of this paper as well as Section 4.1 of FR18. All fitting and analysis in APTB is done via the PINT pulsar timing package (Luo et al. 2019, 2021).[3]

2.1. *JUMPs*

Even though the initial timing ephemeris gives residuals that appear random, there are in fact patterns within each cluster (i.e. single observation or closely spaced sets of observations). PINT has the infrastructure to extract valuable information from these patterns by fitting each cluster for a different arbitrary time offset. More specifically, PINT will find the parameters, including time offsets, such that the *reduced* χ^2 (χ^2_ν, where ν represents the degrees of freedom) of the model is minimized. Allowing PINT to fit a cluster for an arbitrary time offset is called applying a *JUMP* to that cluster. The arbitrary time offset itself may be called a JUMP as well. The reason this method obtains a better solution is that the JUMPs account for unknown phase counts between clusters while the main timing parameters change to account for the phase behavior within each and every cluster. This is particularly useful for finding accurate binary parameters as these can have large effects on short timescales. When JUMPs are applied to multiple clusters, information on the pulsar phase between the clusters is lost. Thus, the JUMPs have to be sequentially removed to obtain a full solution.

To initialize a model, each cluster except one should receive a separate JUMP. At this point, fitting the TOAs for several parameters, like spin frequency (f or $F0$) and a few binary parameters should give a very low χ^2_ν. This χ^2_ν should be very close to unity, and a $\chi^2_\nu \gtrsim 2$ should be cause for alarm—the initial ephemeris is likely too erroneous (e.g. an inaccurate estimate of the initial binary parameters) or the TOA errors have been underestimated. The initial χ^2_ν will be referred to as the base χ^2_ν, or $\chi^2_{\nu,\,\text{base}}$.

2.2. *TOA Scoring*

In this section, we will describe why it is important to rank clusters in terms of importance, and the way that we rank them. When initializing a model, not every cluster can be JUMPed—there needs to be an anchor cluster. This first non-JUMPed cluster will be referred to as the *starting cluster*. As described in Section 5, the starting cluster can have a large impact on whether APTB can discover a solution. It is thus prudent to find the best starting cluster before actually attempting any solution, as running APTB for each and every possible starting cluster would be computationally expensive and likely unnecessary.

We thus adopt a scoring system discussed in Section 4 and Equation 2 of PR22, with only a minor change. This scheme scores each TOA based on how close it is to all other TOAs. A dense group of TOAs will generally all have high scores, while isolated TOAs will ideally be given low scores. Our change to the method described in PR22 is that we apply an index, α, to the weighting scheme such that the score of the ith TOA, n_i, reads

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

where t_j refers to the arrival time of the jth TOA and t_i refers to the arrival time of the TOA being scored. A cluster's score is the highest score of its constituent TOAs.

APTB currently uses $\alpha = 0.3$, although α's best value has not been extensively tested. Moreover, a "best value" is likely highly dependent on the TOA data. APTB uses $\alpha = 0.3$, and not $\alpha = 1$ as in APT, because the scoring system of APT often over-weights particularly dense clusters that are not close enough to other clusters to allow phase connection to proceed.

2.3. *Phase Wraps*

When observing a pulsar, it is necessary to already have good estimates of the relevant parameters. Thus a single observation should correctly identify the relative pulse numbers of its TOAs. In other words, a cluster should be phase connected internally. In the case that a cluster is not phase connected internally, APTB can often correct for the TOAs that break phase connection by tracking the phase during the

Cluster 6_M	Cluster 7_M	Cluster 6_C	Cluster 7_C
2402	3534	2301	3432
2607	3804	2506	3702
2803	–	2702	–

Table 1. Example of the global rotation numbers for the TOAs in Cluster 6 and Cluster 7 respectively. Columns 1 & 2 are the model, M, predictions, and Columns 3 & 4 are the correct, C, predictions. Cluster 6 has 3 TOAs while Cluster 7 only has 2. While the model predicts the wrong pulse numbers for each TOA, the difference in pulse numbers (ΔPN) within each cluster *are* correct (i.e. $2506 - 2301 = 205 = 2607 - 2402$). The ΔPN between Cluster 6_M and Cluster 7_M is 731, while the ΔPN between Cluster 6_C and Cluster 7_C is 730. Thus, Cluster 6 for this model needs a phase wrap of -1 to be corrected.

cluster. The arbitrary definition of zero phase can cause the model to confuse residuals that cross the ±0.5 rotation boundary. This makes these TOAs appear to lack phase connection when they otherwise would be phase connected. APTB corrects these instances by applying appropriate phase wraps on individual TOAs until phase connection within every cluster holds. Applying a phase wrap changes the assigned phase of a single or group of TOAs by an integer step. The correction is done quickly via the `NumPy unwrap` routine.

The initial timing ephemeris will usually not provide phase connection within every individual cluster nor between most of the clusters. Once every cluster has been corrected to be phase-connected within itself, it is necessary to manually change the assigned pulse numbers for each cluster of TOAs so that at least the relative rotation count between two clusters can be compared. The goal of phase connection is to correct the relative rotation count between each and every cluster. The process of manually changing the pulse numbers of an entire cluster from those predicted by the (yet-to-be-correct) model is known as *checking for phase wraps*. See Table 1 for an example. Once phase wraps are used on individual clusters to phase connect the clusters within themselves, phase wraps are then only evaluated between JUMPed, or individually phase-connected, sections of the TOAs in order to map a gap (see Section 2.4).

2.4. *Mapping a Gap*

Checking for phase wraps between clusters can be a computationally expensive process. It can be unclear how many possible phase wraps should be checked. Furthermore, if too many phase wraps are checked, the total number checked for every neighboring pair of TOAs grows quickly. See Section 3.3 for a more detailed explanation of what we call the phase wrap search space, but in short, if an algorithm is not careful, it may be in the process of checking more than 10^{24} phase wraps, and even at a fraction of a second per phase wrap, this would take on the order of 10^{15} years.

Table 1 shows an example where a cluster needs a single phase wrap removed to be correct. Unfortunately, we do not know this *a priori*. To attempt to determine if phase wraps are needed, APTB uses a technique referred to as *mapping a gap* (FR18). Below is a description of the gap-mapping technique used by APTB

Let $\chi_\nu^2(n)$ be the χ_ν^2 after applying a phase wrap of n. If a phase wrap of m minimizes $\chi_\nu^2(n)$, then a Taylor series of $\chi_\nu^2(n)$ centered around m has no linear n term and so the n^2 term is the leading non-constant term. Therefore, the $\chi_\nu^2(n)$ has a quadratic dependence on the phase wrap when n is close to m (see also Figure 5 of FR18). Finding the vertex in phase wrap-$\chi_\nu^2(n)$ space gives m, which represents the best phase wrap according to the best, but not necessarily fully correct, model. The parabola can be defined by sampling the $\chi_\nu^2(n)$ for three different phase wraps (i.e. by sampling three phase wrap-$\chi_\nu^2(n)$ points) and the phase wrap of the vertex is given by the closest integer (i.e. $\mathrm{round}[m]$) to

$$ m = \frac{b}{2} \frac{\chi_\nu^2(-b) - \chi_\nu^2(b)}{\chi_\nu^2(b) + \chi_\nu^2(-b) - 2\chi_\nu^2(0)}. \tag{2}$$

This is the formula for calculating the vertex x-coordinate of the parabola $f(x) = a_2 x^2 + a_1 x + a_0$ by sampling the values of $f(b)$, $f(-b)$, and $f(0)$. APTB uses $b = 5$, as do FR18, though this can lead to an error if $b = 5$ gives a nonphysical parameter value, like a negative value of the projected semi-major axis ($a \sin i$). If this occurs, $b = 4$ is attempted, repeating lower b values until either the vertex is successfully calculated or $b = 0$. If the $b = 0$ iteration is reached, we cannot solve for the vertex of the parabola and the model is likely too poor to successfully phase connect, and so APTB will exit. A maximum $b = 5$ is chosen because $b \lesssim 3$ would let small variations in the χ_ν^2, such as from TOA noise, greatly affect m in Equation 2. Additionally, $b \gtrsim 7$ would likely involve sampling phase wraps far from the phase wrap minimum—as the minimum is commonly at a phase wrap of order unity—weakening the quadratic-dependence assumption.

As for DRACULA, the gap-mapping method can only be successful if the TOAs are accurate to within their predicted error. If even a few TOAs are entirely erroneous, APTB will be fitting the parameters to compensate for fictitious differences in phase. Even worse, incorrect phase wraps will be treated as correct, and any model stemming from a model with even a single incorrect phase wrap can never be entirely phase connected when additional correct TOAs are included. Before phase connection is complete, it is nearly impossible to determine outliers because the phase connection process is sensitive enough to require the assumption of correct data. Therefore, it is prudent that APTB is given correct TOAs that are as accurate as possible, with special attention given to removing TOAs with high predicted error. APTB cannot currently address the cases where some TOAs are in error. A potential improvement to the algorithm would be to have branches (Section 3) investigate the removal of TOAs or entire clusters.

3. SOLUTION TREE

We begin this discussion by comparing pulsar timing to exploring a tree-like structure. The root or trunk of the tree is the starting model and different phase wrap decisions form branches. As different phase wraps are applied to attempt phase connection, more branches are created. This can lead to an exponential time complexity ($O(2^n)$ where n is the number of clusters) if the algorithm accepts too many possible phase wrap options (Section 3.3). Our approach assumes that several incorrect branches give a large χ_ν^2 early on in the phase-connection process so that these branches can be quickly discarded.

The best phase wrap in the short term is not necessarily the best phase wrap in the long term. In many cases, several possible phase wraps in a gap can give a χ_ν^2 that passes the pruning condition (see Section 3.2). The χ_ν^2 for adjacent phase wraps can also be very similar, so even deciding the "best" phase wrap can be somewhat arbitrary, or at least weakly justified. Furthermore, we know our best-guess model is incorrect on some level so there is no reason to only choose the lowest χ_ν^2 phase wrap when this χ_ν^2 is based on a not-yet correct model. It thus becomes necessary to explore all acceptable branches, where acceptable is defined as having a χ_ν^2 below a predetermined value known as the pruning condition. We used the Python package TREELIB[4] to manage traversal and pruning of the tree and its requisite bookkeeping, though other Python data tree libraries could serve the same purpose.

3.1. *Branching*

Going down all acceptable branches is like trying to produce a tree-like structure that is actively forming with every decision made. The tree's structure is only revealed by exploring it. APTB creates the tree structure by mapping each gap and testing phase wraps. Phase wraps that fail the pruning condition result in that branch being cut. Phase wraps that pass are ranked based on lowest to highest χ_ν^2. The best phase wrap (lowest χ_ν^2) in the short term is explored first, with its own branches being explored next. Thus, this is a depth-first search (see Figure 2).

3.2. *Pruning*

While exploring several branches is often needed to phase-connect pulsars with limited data, it is important to know when a dead end is reached. In pulsar timing, there are an
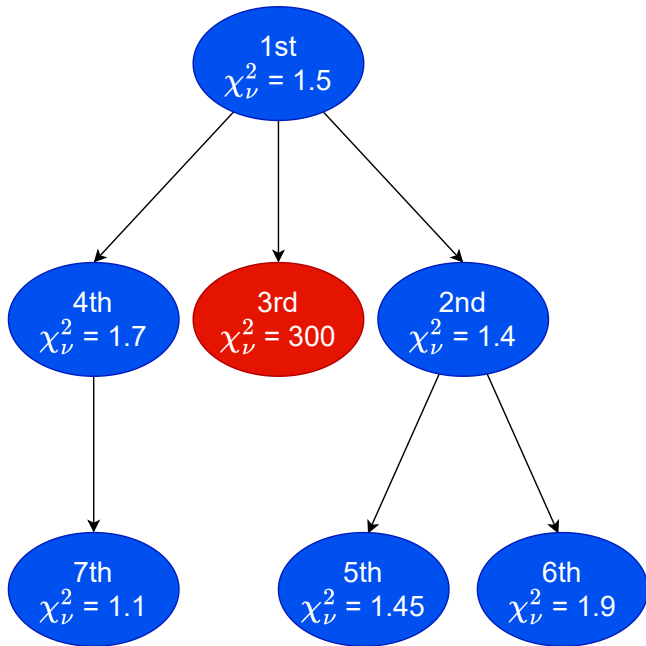
---

[4] https://pypi.org/project/treelib/

Figure 2. Flowchart of APTB's branch searching priority. Even though the bottom left model has the lowest χ^2_ν, its parent prevented it from being immediately explored. Branching is based on phase wrap decisions so the 2nd, 3rd, and 4th models represent different phase wrap decisions that phase connect one more cluster than the 1st model. For example, the 2nd model could result from a phase wrap of 0, the 3rd model from a phase wrap of 1, and the 4th model from a phase wrap of -1. Keep in mind no model 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.

infinite number of options available so we have to be clever in deciding which branches, or phase wraps, are in fact dead ends. Thus a branch must be pruned when it becomes clear that it cannot lead to the correct solution.

When it is time for a cluster to be unJUMPed, the gap-mapping technique is applied. Again, gap-mapping allows APTB to know a reasonably good phase wrap to start with. The χ^2_ν is calculated for the vertex point as well as for the phase wraps of ± 1 from the vertex. The minimum χ^2_ν of these three phase wraps, now referred to as PW_{calc}, must be lower than the pruning condition. The pruning condition, PC, used by APTB is $PC = \chi^2_{\nu,\,\text{base}} + 1$. If $PW_{\text{calc}} \geq PC$, then the F-test for model comparison is done on the current model vs. the current model plus a new parameter. The F-test provides a way to compare two models that have different degrees of freedom. We refer to Section 3 of PR22 for a brief discussion on the F-test for model comparison. This model is given one more chance and the gap-mapping technique is applied one more time, with PW_{calc} calculated again. If $PW_{\text{calc}} \geq PC$ still, this branch is pruned. This model is given a second

chance because in some instances evaluating the F-test with different parameters should be done before mapping the gap.

When a branch is created, certain parameters are checked to see if they are physically relevant. For instance, if $a \sin i$ (the projected semi-major axis of the orbit) is negative, then the branch is pruned. Optionally, a branch is pruned if $\dot{f}$ ($F1$) becomes positive, though this is not recommended as $F1$ could become negative in a later model stemming from this branch. However, a pulsar can be observed to have a non-intrinsic positive $F1$ if it has a negative radial acceleration. This is common for pulsars in globular clusters due to the gravitational potential of the clusters. In fact, two of the pulsars discussed in Section 6, PSRs J1748−2446aq and J1748−2446at, have a positive $F1$. Also optionally, if the right ascension (R.A.) or declination (decl.) leaves a set boundary, the branch is pruned. No other pruning mechanisms are implemented currently.

3.3. *Phase Wrap Search Space*

We pause the explanation of the algorithm to elaborate on why and when APTB may succeed or fail. The models that APTB explores based on its different phase wrap decisions all comprise the phase wrap search space (PWSS). The size of this space can be the dominant factor in determining whether APTB will succeed or not. In the case of Figure 2, the size of the PWSS is 7 models. Generally speaking, the size of the PWSS, S, is

$$S = \sum_{d=0}^{N_{\text{clusters}}} W(d), \qquad (3)$$

where N_{clusters} is the number of clusters and $W(d)$ is the number of models, or width, of the solution tree at a depth d. The depth refers to the number of clusters already phase connected for a given model minus one. In Figure 2, $W(0) = 1$ and $W(1) = W(2) = 3$. If APTB takes τ seconds on average per model, then the total run time would be τS.

The nature of $W(d)$ marks the boundary between a solvable and unsolvable pulsar using our algorithm. We will start with just one example where APTB would fail. If every gap has two plausible phase wraps (two branches), then $W(d) = 2^d$, implying $S \geq 2^d$. For the case of PSR J1748−2446aq (Section 6), $\tau \sim 10$ s and $N_{\text{clusters}} \sim 80$. Therefore, the total run time would be of order $10 \cdot 2^{80}$ s $\sim 10^{17}$ yr. Clearly, APTB would be unsuccessful for PSR J1748−2446aq if it needed to check two branches for every gap.

APTB can be successful due to most pulsars following a different $W(d)$. The initial model is precise enough to phase-connect TOAs within an observation but not between observations (i.e. clusters). Therefore, $W(1)$ can be quite high, and $W(2)$ through roughly $W(5)$ can become exponentially higher. Eventually, a model has phase-connected enough clusters to be much more precise in its predictive power and so admits exponentially fewer possible branches,

significantly lowering the width at higher depths. This all suggests we adopt

$$W(d) \sim \left\{ \begin{array}{ll} r_1^d, & \text{if } d \leq d_0 \\ r_1^{d_0} r_2^{(d-d_0)}, & \text{if } d > d_0 \end{array} \right\}, \qquad (4)$$

where r_1 and r_2 are the growth and decay rates, respectively, and dictate $W(d)$ based on the cutoff depth, d_0. The values of r_1, r_2, and d_0 will determine if APTB can solve a pulsar. Non-integer values of $W(d)$ can be interpreted as a characteristic value. Assuming $W(d \lesssim d_0) \gg W(d \gg d_0)$, we can simplify further to $S \sim \sum_{d=0}^{d_0} W(d) \sim r_1^{d_0}$. Therefore if either r_1 or d_0 is too large, S will also be too large. r_1 represents how quickly the model will branch until a depth of d_0 is reached. Pulsars with bad data (e.g. due to erroneous TOAs or low cluster density) will initially accept many phase wraps (increasing r_1) and could continue to accept them for longer (increasing d_0).

While both human and computer-based timing rely on the search space not becoming too large, a computer is able to probe a much larger search space. This is helpful in the cases where the cutoff depth allows for S no larger than on the order of hundreds of models to a few thousand models, but not when the search space balloons to tens of thousands of models or more. A human could take days to search $S \sim 100$ models but our algorithm can take minutes or hours and is fully automated. This is not to mention that APTB would be more systematic than a human, searching for phase wraps in the same way every time. When the search space is in the tens of models, then both humans and APTB would find no issue, but APTB would still likely phase connect faster.

This is all to say that while APTB can solve trickier pulsars than a human could solve, APTB can still fail even when functioning as intended. Thinking of our algorithm with the PWSS in mind can give a first-level analysis of the limits of our algorithm.

4. BINARY MODELS

It is important to choose the appropriate binary model as each model has its own unique benefits in terms of not only enabling the possibility of phase connection but also the overall computational efficiency of phase connection. Thus, the binary model should already be selected by the user, with proper estimates of the binary parameters established. Currently, APTB can handle the ELL1 (Lange et al. 2001), BT (Blandford & Teukolsky 1976), and DD (Damour & Deruelle 1985, 1986) models. Implementing additional binary models would be straightforward.

4.1. *Nearly Circular Orbits*

Most binary MSPs have nearly circular orbits where is it difficult to measure the longitude of periastron (ω) and epoch of periastron passage (T_0). This is because the TOAs do not

clearly indicate the location of periastron, and thus the epoch of periastron is not clearly indicated either (Lange et al. 2001). The ELL1 model was designed for nearly circular orbits to address this difficulty by measuring the orbital phase with respect to the epoch of ascending node (T_{asc}) rather than T_0. The ELL1 model is a Newtonian binary model that neglects eccentricity (e) terms of order e^2 or higher, although most timing software have improved the accuracy of the ELL1 model by adding e^2 or even e^3 terms. Therefore, this model describes residuals that follow a slightly perturbed sinusoid as a function of the orbital phase. Part of APTB's initialization is to not start fitting for $\epsilon_1 = e \sin \omega$ and $\epsilon_2 = e \cos \omega$ immediately. ϵ_1 and ϵ_2 can only be fit after the unJUMPed clusters span 5 times the binary orbital period (P_b), though this can be changed by the user. In contrast, the model should immediately begin fitting for P_b, T_{asc}, and $a \sin i$ in order to correct for orbital effects within every cluster. Also, for this reason, these parameters should be known to a satisfactory precision beforehand. Notably, when the F-test processes are conducted (Section 3.2), both ϵ_1 and ϵ_2 are tested together. If they significantly help the model they are added to the model together. In many cases, the eccentricity is negligible so assuming $\epsilon_1 = \epsilon_2 = 0$ is often sufficient for phase connection.

It should be noted that while using the ELL1 model, APTB does admit post-Keplerian parameters such as the time derivative of the binary period ($\dot{P}_b$). These are very rarely required for phase connection on short timescales, though, and are usually included after phase connection has already been achieved. We briefly discuss the use of $\dot{P}_b$ in the timing of PSR J1748−2446ar in Section 6. Post-Keplerian parameters are only handled if the user explicitly includes them in the initial parameter file.

4.2. *Non-Circular Orbits*

Though less common, some binary pulsars are elliptical enough to require binary models that make no assumptions about the eccentricity. One such model is the BT model, a Newtonian model. If the eccentricity is important for phase connection purposes, this binary model should be used. APTB will begin fitting for all major binary parameters immediately, and therefore no testing via the F-test is done on these parameters. The base parameters of the BT model are P_b, $a \sin i$, T_0, e, and ω. Like in the ELL1 model, APTB can also handle post-Keplerian parameters provided the user includes them in the initial parameter file.

Another non-circular orbit model is the DD model, a post-Newtonian approximation relativistic model that is theory-independent. This model is intended for pulsars with large eccentricities and measurable secular and periodic post-Newtonian effects. Similar to the BT model, all major parameters will be fit for immediately. These parameters are

P_b, $a \sin i$, T_0, e, ω, and the time derivative of the longitude of periastron ($\dot{\omega}$).

5. ALGORITHM

APTB is an algorithm that uses JUMPs, the F-test, gap-mapping, and a data tree structure in order to phase connect isolated and, more specifically, binary pulsars. We strongly recommend using the flowchart given in Figure 3 to help guide the rest of this section. Before diving into the specifics of the algorithm, it is helpful to describe it on a higher level. The best starting cluster is selected and APTB will attempt to remove JUMPs from more and more clusters to eventually phase connect every TOA. To do this, adjacent clusters to the starting cluster are unJUMPed. We know the model is incorrect on some level, yet should give almost the correct relative pulse number difference between clusters. To verify this, we determine the χ^2_ν of various phase wraps. This should have a parabolic dependence assuming the model is not too incorrect. Then the most plausible phase wraps can be investigated. Using a depth-first search technique, APTB will investigate all branches. If the initial starting cluster yields no acceptable solutions, APTB will try other starting clusters.

5.1. *Algorithm In Depth*

The model is initialized by applying JUMPs to every cluster except one. APTB then ensures that each cluster is phase connected internally. The highest-scoring cluster, as defined by Equation 1, is chosen as the starting cluster and is the only cluster not JUMPed. If the χ^2_ν is above three at this point, there may be a problem that APTB cannot fix. APTB will attempt to fix this by phase connecting and fitting two more times, but this is likely to fail. Either the starting timing model is too poor, the TOAs are inaccurate, or the TOA uncertainties are underestimated. In any case, the user must do something on their end before proceeding. This can include fixing one of the above problems or overriding this warning (see Section 5.2).

Provided the model has been properly initialized, the main algorithm can begin. The starting cluster is the first member of the *unJUMPed clusters*. The cluster closest in time to the unJUMPed clusters is unJUMPed and therefore added to this group. The number of pulsar rotations between the closest cluster and the other members is not known, so the phase wraps between the closest cluster and the previously phase-connected data are checked according to the gap-mapping methodology (Section 2.4). Gap-mapping (ideally) finds the minimum χ^2_ν for any particular phase wrap, but another phase wrap may indeed prove to be the correct one. For this reason, APTB incrementally checks the phase wraps $1, 2, 3, ..., N_{\text{below}}$ below the best phase wrap until the model fails the pruning condition at a phase wrap of $N_{\text{below}} + 1$ below the best phase wrap. The same is repeated for $1, 2, 3, ..., N_{\text{above}}$ above the

best phase wrap, with the model failing the pruning condition at the $N_{\text{above}} + 1$ phase wrap above the best phase wrap. The phase wraps between N_{below} and N_{above}, inclusive, are acceptable models that represent branches on the solution tree.

All acceptable solutions are stored at this point, and attached to the parent model that created them. The next used model is determined by the branch search hierarchy (see Section 3.1 and Figure 2). This next model is then fit for new parameters if they significantly improve the model. As in PR22, an F-test value of 0.005 or lower is deemed a significant improvement (i.e. the probability that the model fit improvement could be due to noise is <0.005). Not every non-fit parameter is tested. The right ascension and declination can be tested after the unJUMPed clusters span at least T_{RA} and $1.3 \cdot T_{\text{RA}}$, respectively, where

$$T_{\text{RA}} = \left(-\frac{30}{700} \cdot \frac{f}{\text{Hz}} + 40 \right) \text{ days}. \qquad (5)$$

The functional form of T_{RA} is such that T_{RA} depends linearly on the pulsar's spin frequency, $T_{\text{RA}}(f = 700 \text{ Hz}) = 10 \text{ days}$, and the maximum cluster span until the right ascension is tested is 40 days (i.e. $\max[T_{\text{RA}}] = 40 \text{ days}$). The linear dependence condition is motivated by the fact that the sky position becomes important quickly for pulsars with higher TOA precision, which is usually directly proportional to the spin frequency. The three requirements for Equation 5 have not been derived from first principles, but rather from experience phase connecting many pulsars manually. $F1$ can be tested after the unJUMPed clusters span is long enough that a typical $F1$ would cause the residuals to exceed ± 0.35 in phase. A typical $F1$ is estimated based on the pulsar's spin frequency and rough placement in the P-$\dot{P}$ diagram (see e.g. Figure 6.3 of Condon & Ransom (2016)).

Currently, APTB has the capability to conduct an F-test and fit for $\ddot{f}$ ($F2$), but the user has to specify a minimum unJUMPed cluster time span since the default is not to test for $F2$. We choose to omit $F2$ by default because $F2$ usually only becomes significant in time spans longer than are necessary for phase connection. Furthermore, typical $F2$ values vary widely, and estimating when $F2$ should be fit is difficult; a wrong guess may prevent the correct solution from being found due to possible covariances with the other parameters. In most cases, $F2$ can be ignored until after a successful phase connection, and then manually fit, as we did for PSR J1748−2446aq (Section 6).

After the appropriate F-tests and new parameter additions, the model is then brought back to the beginning of the algorithm, as all model selection from the solution tree is done after checking for phase wraps. The first step is, again, to unJUMP the closest unJUMPed cluster, and so on.

Once every TOA has been unJUMPed, if $\chi^2_\nu < 10$ then the model is deemed a possible solution and saved. APTB

Figure 3. Flowchart of APTB's logic.

then moves to the next starting cluster to see if another (or the same) solution arises.

In some cases, every branch is pruned before finding a solution. This means APTB has failed to find the solution with this particular starting cluster. The algorithm will attempt to run again with the second-best starting cluster and will keep selecting starting clusters until the top five (default) clusters by score have been attempted. While we hope to make APTB robust enough to find the solution regardless of a starting point, some assumptions break down when the model is too poor or the data is too sparse. When searching for correct phase wraps, APTB uses the quadratic dependence of the χ^2_ν to efficiently find the minimum. If the minimum phase wrap is too large or the model is too poor, this quadratic dependence falls apart quickly. This could pin every early-on model into a local minimum rather than the global minimum. If this local minimum is too wide, as in the parabola covers a large number of phase wraps, or if there are too many local minima, then APTB may not find any solutions or a very large number of wrong solutions.

APTB will store all explored models on the user's local disk. These files are crucial in the case APTB is very close to finding a solution but failed due to unforeseen circumstances. For example, if the model requires $F2$ or the derivative of the binary orbital period, it may have pruned the correct branch. By finding this almost-complete model, the remaining few steps can be manually completed quickly. For more details on the files saved and how to use them, see the APTB User Guide.

5.2. *Optional Features*

APTB comes equipped with several ways to modify its behavior from the command line. We will describe the most useful modifications, with the rest being listed and briefly described in the APTB User Guide and the Python file on GitHub. By default, APTB will check for phase wraps and also use the solution tree structure. These can separately be turned off by including `--no-check_phase_wraps` and `--no-branches`, respectively, in the command line. But note that branching only works if phase wrap checking is on. If phase wrap checking is turned off, then APTB will assume that the cluster that just had its JUMP removed has a phase wrap of zero with the unJUMPed clusters. In other words, at each step, APTB assumes the model correctly predicts the relative pulse numbers after removing a JUMP.

If phase wrap checking is on but branching is not, then APTB will map the gap and find the phase wrap with the lowest χ^2_ν, but will not pursue nearby phase wraps that are also acceptable. For some pulsars, this is sufficient to time them successfully. While APTB is really meant to excel in the case where both phase wrap checking and branching are

required to time the pulsar, the algorithm can also easily and quickly time the less challenging case.

When APTB finds a potential solution, there may be other unexplored branches in the solution tree that may yield other potential solutions. By default, APTB will explore these branches to rule out, or discover, any degenerate solutions. This can be turned off, but doing so can have similar effects to turning off branching in that the correct solution may not be initially the best one. The advantage to stopping APTB from pursuing branches after finding a potential solution is that there may be a large number of potential solutions (tens or hundreds) which can consume much more computing time and storage than anticipated.

Some command line options affect the pruning condition. By default, if $\chi^2_{\nu,\,\text{base}} > 3$ then the algorithm will not proceed. This can be overridden if the user thinks APTB should try to time this pulsar anyway. The pruning condition by default is set to $\chi^2_{\nu,\,\text{base}} + 1$ but the user may specify their own value. When APTB can fit for R.A., decl., $F1$, and certain binary parameters like ϵ_1 and ϵ_2 can be adjusted manually. Lastly, a rectangle in R.A.-decl. space can be specified. A model will be pruned if its astrometric position leaves this rectangle.

6. RESULTS

One important test of APTB is whether it can time and solve a pulsar that has never been phase-connected before. Indeed, we demonstrate the capability of APTB by showing how it was first in determining the phase-connected timing solutions for the newly discovered PSRs J1748−2446aq and J1748−2446at. We also show that it was able to verify phase-connected timing solutions for the PSRs J1824−2452N and J1748−2446ar.

PSR J1748−2446aq, Ter5aq hereafter, was discovered in February of 2023 (Padmanabh et al., in preparation) and is the forty-second pulsar discovered in Terzan 5.[5] Terzan 5, a globular cluster near the galactic bulge, is home to the largest globular cluster pulsar population (Martsen et al. 2022), with several recent new discoveries. Ter5aq has 86 clusters available spanning 13 years, which gives an average cluster density of roughly 0.018 clusters/day or 1 observation every 55 days. Most pulsars have a much higher observation cadence. Moreover, no observations in the TOA data provided to APTB were dedicated specifically for phase connection purposes. That is, APTB did not require a grouping of clusters with a significantly higher cluster density than the average. With APTB phase connecting Ter5aq first, this verifies that APTB can time low cluster density pulsars. We show the final residual plot in Figure 4. Ter5aq is a millisecond pulsar with a 13 ms spin period, and is in a binary system with an orbital period of 0.12 days and a 0.026 light-second projected semimajor axis.
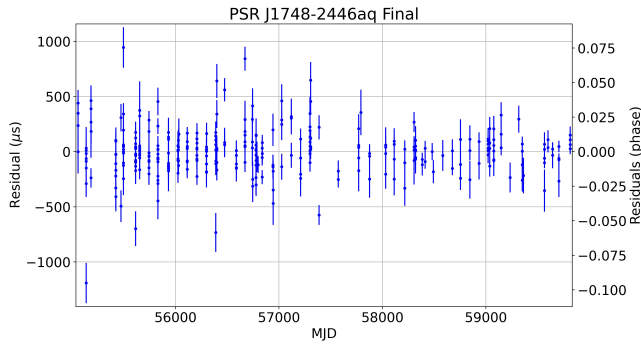

Figure 4. TOA residuals for PSR J1748−2446aq after APTB's successful phase connection.

The full solution from APTB, including additional data, will be published in Padmanabh et al. (in preparation).

To reinforce the difficulty a human would have in timing Ter5aq with the available data, we include Figure 5, which schematizes the phase wrap decisions APTB made. APTB took 416 iterations to complete its search, and 106 iterations of the first 109 iterations were all on branches leading to a dead end. It was not until the 110th iteration that APTB was on the correct branch and stayed on it. The large phase wrap search space means APTB took 5.0 hours, or 1.0 hours on average per starting cluster. The first starting cluster that was successful took 1.5 hours. This may seem like a long runtime, but a human would likely take much longer and give up from frustration after many incorrect attempts. Because $F2$ was excluded from being fit for, APTB only reached a depth of 30, instead of the full depth of 86 clusters. Therefore, APTB was not confident it found the correct solution, but upon inspection of the solution tree, where post-fit timing residuals are plotted and saved, it became obvious that the phase connection was near complete. Even after reaching a depth of 30, APTB continued to look for more solutions to ensure the solution was unambiguous. While Ter5aq was eventually timed independently by a human, it is easy to imagine a pulsar with even less dense observations or a smaller number of them. In this case, using the automatic capabilities of APTB might be the only option to phase-connect that pulsar.

PSR J1748−2446at, Ter5at hereafter, was discovered in April 2023 (Padmanabh et al., in prep) making it the forty-fifth Terzan 5 pulsar (PSR J1748−2446ar, see below, was discovered before Ter5at). Ter5at is a black widow system, a type of binary pulsar with a companion of only a few hundredths of a solar mass (Shahbaz et al. 2017). While also eventually phase-connected by a human, APTB determined the proper phase connection first. The 2.2 ms spin period is significantly faster than that of Ter5aq, and its orbital period and projected semimajor axis are 0.22 days and 0.10 light-second, respectively. Again, the full solution will be published in Padmanabh et al. (in preparation). The cluster

density is 0.016 clusters/day (an observation every 62 days), and APTB's runtime was 5.3 hr. The first starting cluster did not lead to a solution, but the other four did. The second starting cluster found the solution the quickest and was on the correct phase wrap branch immediately. As in the case of solving Ter5aq, APTB reached a tree depth where the residual effects of $F2$ became important, in which case APTB thought it reached a dead end. This was easily spotted in the solution tree, and the process of finishing the phase connection was straightforward.

APTB can also time pulsars with data spans shorter than 14 years, provided the cluster density is higher. This is exemplified by the successful timing of PSR J1824−2452N, M28N hereafter. M28N was discovered by Douglas et al. (2022) and is the fourteenth pulsar discovered in the globular cluster M28. Douglas et al. (2022) describe M28N as a Black Widow MSP with a binary period of 4.76 hr. APTB was given only about 3 years of TOAs with a cluster density of roughly 0.035 clusters/day (an observation every 29 days), roughly twice that of Ter5aq. As is the case for pulsars with high TOA densities, APTB struggled less to time M28N but was still on a dead-end branch for its first 5 iterations. This pulsar was fit for only the first 900 days because a longer timescale makes $F2$ non-negligible.

APTB has been able to time several more real pulsars with known phase-connection, but we will only mention one more instructive example. PSR J1748−2446ar (Ter5ar hereafter) is the forty-third pulsar discovered in Terzan 5 (Padmanabh et al., in prep). This is a redback pulsar, a type of pulsar characterized by having a stellar companion with a few tenths of a solar mass (Shahbaz et al. 2017). A redback pulsar's wind often ablates its companion, creating intra-binary material that eclipses the pulsar's radio pulses every orbit. The material, as well as the non-degenerate nature of the companion star, can cause years-long timescale perturbations in the residuals. The specific patterns of these effects appear to be largely random, making modeling and predicting pulses far into the future very tedious or impossible. As such, the $\chi^2_{\nu,\text{base}} \sim 8$ is rather high, and this was after removing TOAs that were clearly erroneous. The high $\chi^2_{\nu,\text{base}}$ can be explained by the fact that a redback system's orbital variability can be significant over even a couple of years. For this reason, APTB was only given roughly the first year of data on Ter5ar. The derivative of the orbital period ($\dot{P}_b$) can be important on this timescale, with $\ddot{P}_b$ and $\dddot{P}_b$ both becoming crucial on any timescale exceeding a year due to the short binary period and small $a \sin i$. In spite of these difficulties, APTB found the solution, and the correct branch was the first branch it investigated. APTB's success is likely thanks to the relatively high cluster density of 0.071 clusters/day (an observation every 14 days). Owing to the nature of the binary system, the final $\chi^2_{\nu} = 10.1$, which is high but comparable to the $\chi^2_{\nu,\text{base}}$. No other plausible
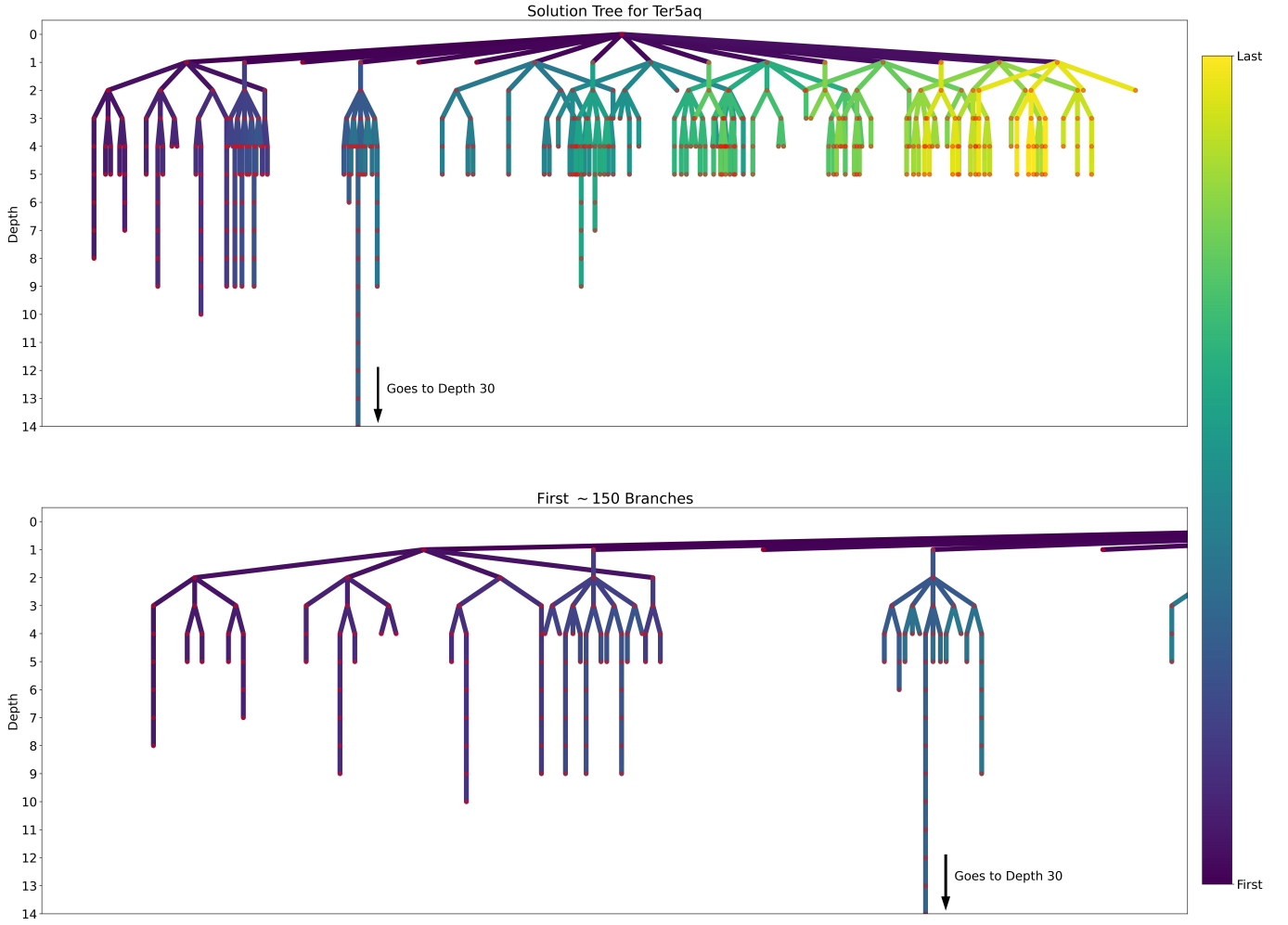
Figure 5. A schematic of the solution tree APTB investigated in order to time Ter5aq. The branching order that APTB took is qualitatively described by the color bar. The dark purple lines show branches that APTB investigated first while the light green and yellow lines show branches APTB investigated last. The depth of the solution tree is equivalent to the number of clusters that had been unJUMPed at that stage minus one. The green and yellow branches were investigated heavily, but no meaningful solution was found. The dark blue-turquoise branch that goes past a depth of 14 actually ends at a (not shown) depth of 30. This emphasizes that the depth-30 branch in fact leads to the unique solution. The lower panel shows only the first ~150 branches investigated.

solution was found, and in spite of the abnormally high χ_ν^2, the found solution properly phase connects the next year of data.

7. CONCLUSION

We have created an algorithm, APTB, that can phase-connect isolated and binary pulsars. Due to APTB's robust branch-searching structure, it was able to time and solve PSRs J1748−2446aq and J1748−2446at before any human could. APTB also timed several other known pulsars, including PSRs J1748−2446ar and J1824−2452N. A standard successful run takes on the order of tens of minutes to several hours. In cases where APTB is unsuccessful in phase connection, the process can take much longer as the phase wrap search space expands exponentially.

Like other pulsar timing algorithms, APTB allows a computer to make the same decisions that a human would. Even in the case of hour-long run times, this is of no concern to the pulsar astronomer as APTB is fully automated. Therefore, our algorithm stands as an example of how current and future astronomers can grapple with ever-enlarging data sets. Pulsars discovered in globular clusters benefit from abundant archival data that an automated algorithm like APTB can easily handle, saving manual work-hours. Moreoever, a pulsar timing algorithm can become a necessity to successfully phase connect a binary pulsar when its data is too sparse. The increase in pulsar discoveries without a corresponding increase in available telescope time will only make sparse-data binaries more prevalent. While there already exists pulsar timing algorithms that can time binary pulsars, like DRACULA, APTB has the im-

portant advantage of being fully automated and therefore is much more suitable for binaries with sparser data. Lastly, APTB is currently maintained, open to improvements, and is written exclusively in the modern and popular Python coding language. Several improvements, including support for new binary models and more options for weighting clusters, among others, are planned.

Binary pulsars are some of the most scientifically interesting pulsar systems, and phase connecting them is crucial for determining their scientific potential. Tools like APTB allow us to accomplish that with less telescope time and less human effort.

REFERENCES

Agazie, G., Anumarlapudi, A., Archibald, A. M., et al. 2023, ApJL, 951, L8

Antoniadis, J., Freire, P. C., Wex, N., et al. 2013, Sci, 340, 1233232

Antoniadis, J., Arumugam, P., Arumugam, S., et al. 2023, A&A, 678, A50

Blandford, R., & Teukolsky, S. A. 1976, ApJ, 205, 580

Condon, J. J., & Ransom, S. M. 2016, Essential radio astronomy, Vol. 2 (Princeton University Press)

Cromartie, H. T., Fonseca, E., Ransom, S. M., et al. 2020, NatAs, 4, 72

Damour, T., & Deruelle, N. 1985, AIHPA, 43, 107

—. 1986, AIHPA, 44, 263

Demorest, P. B., Pennucci, T., Ransom, S., Roberts, M., & Hessels, J. 2010, Natur, 467, 1081

Douglas, A., Padmanabh, P. V., Ransom, S. M., et al. 2022, ApJ, 927, 126

Einstein, A. 1915, AbhKP, 25, 778

Freire, P. C., & Ridolfi, A. 2018, MNRAS, 476, 4794

Hewish, A., Bell, S. J., Pilkington, J. D., Frederick Scott, P., & Collins, R. A. 1968, Natur, 217, 709

Hobbs, G., Archibald, A., Arzoumanian, Z., et al. 2010, CQGra, 27, 084013

Kramer, M., Stairs, I., Manchester, R., et al. 2021, PhRvX, 11, 041050

Lange, C., Camilo, F., Wex, N., et al. 2001, MNRAS, 326, 274

Luo, J., Ransom, S., Demorest, P., et al. 2019, PINT: High-precision pulsar timing analysis package, Astrophysics Source Code Library, record ascl:1902.007

—. 2021, ApJ, 911, 45

Martsen, A. R., Ransom, S. M., DeCesar, M. E., et al. 2022, ApJ, 941, 22

Phillips, C., & Ransom, S. 2022, AJ, 163, 84

Reardon, D. J., Zic, A., Shannon, R. M., et al. 2023, ApJ, 951, L6

Shahbaz, T., Linares, M., & Breton, R. 2017, MNRAS, 472, 4287