Graph theory and pulsar astronomy tie the knot: the use of labeled graph kernels in exploring the pulsar $\dot{P}$-P diagram

Jacques Maritz[1*], Elizabeth Maritz[2] and Pieter Meintjes[1]

[1]Physics Department, University of the Free State, 9300, South Africa
[2]Mathematics Department, University of the Free State, 9300, South Africa

E-mail: *maritzjm@ufs.ac.za

Abstract. In this paper we explore the dependency of pulsar population structures (seen in $\dot{P}$-P diagram) on the measurable characteristics of pulsars. We implement graph kernels for this investigation and it forms part of structure mining which is a domain of learning on structured data objects in machine learning. Among others, we implement one of the most powerful graph kernels that is based on random walks, and has been successfully applied to data mining projects in the field of astronomy. With instruments such as the SKA coming online in the near future, the quest continues to search for relationships between the different pulsar populations.

1. Introduction

Pulsars can be considered as stable cosmic clocks that serve the purpose of testing fundamental astrophysical theories and advancing computing technology [1]. However, the model describing the evolutionary path of pulsars on the $\dot{P}$-P diagram is incomplete and continuously undergoes upgrades in both complexity and completeness. The $\dot{P}$-P diagram describes the evolution of pulsars in the parameter space described by their spin-period (P), spin-down rate ($\dot{P}$), magnetic field (B) and characteristic spin-down age (τ_c), see Figure 1. The $\dot{P}$-P diagram is mostly populated by normal pulsars from where they evolve beyond the graveyard line to be reborn as isolated or binary millisecond pulsar systems (recycled pulsars).

The $\dot{P}$-P diagram is not just being populated via radio observations, but due to the fact that pulsars produce pulsed-emission across a large part of the electromagnetic spectrum, including optical, X ray and gamma ray, we can gather a wealth of information to try and understand the evolutionary path of pulsars on the $\dot{P}$-P diagram and clustering can accelerate this aim.

Graph clustering techniques are being used in machine learning and social network classification [3]. Specific graph kernels are also being used in photon event tracking seen in the latest Pass-8 Fermi data (http://fermi.gsfc.nasa.gov/ssc/data/). The new technique ultimately uses the three dimensional calorimeter readout and implements a Minimum Spanning Tree (MST) construction to minimize ghost signals and improve the event reconstruction process at GeV levels [4]. In earlier work, [5] proposed using the Minimum Spanning Tree algorithm to classify sources in gamma-ray bi-dimensional images where a photon's (or events) celestial coordinates represent a node in the graph and the edge is weighted by the angular distance

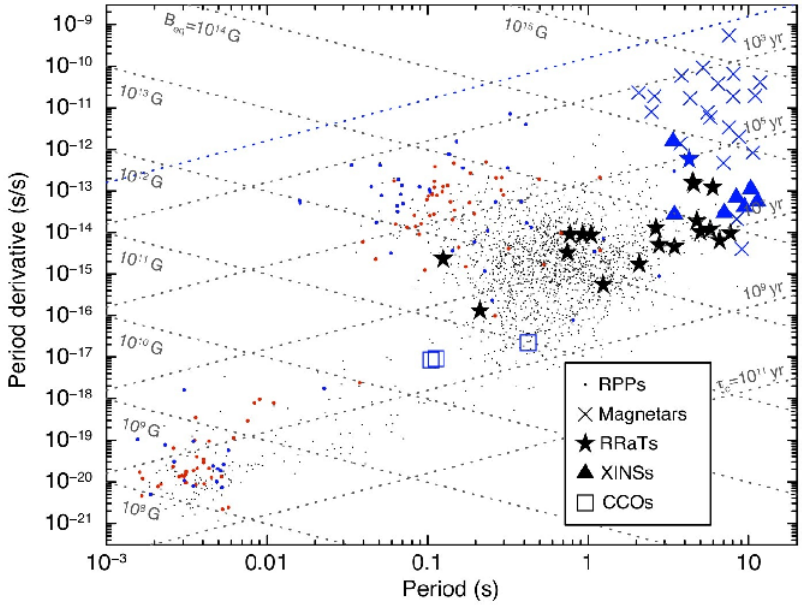


Figure 1. $\dot{P}$-P diagram illustrating the placement of isolated neutron star classes according to their characteristics, adopted from [2].

between the nodes. They applied this algorithm successfully to EGRET data of the Virgo field and detected several sources.

It needs to be stressed that graph kernels are highly application specific, i.e. when considering clustering applications on data we can use a specific kernel called the graph clustering kernel. In the next section we investigate the mechanism of graph clustering, after which we consider the application of graph clustering on the $\dot{P}$-P diagram via unsupervised machine learning.

2. Machine learning

There are two types of machine learning supervised and non-supervised. There are two main trends in machine learning. The first is the classification of objects with respect to a certain characteristic that they might or might not possess. Here we would start with a set of objects with predetermined outputs, i.e. a set of objects that are already classified in terms of either having a certain property, or not. The machine then uses this information, compares these objects and 'learns' what it is that makes them fall into one category or the other in terms of their similarity. The idea is to classify any new object according to what has been learned. The ultimate aim for classification is therefore *prediction*. The second is used for data that does not have clear-cut classifications, or perhaps we do not know what it is that we want to search for in terms of a classification. This then leads to clustering, which is simply grouping together elements that seem to demonstrate similar characteristics, without having to go into too much detail of what those characteristics are. The aim here is therefore *description*.

2.1. Graph kernels

We briefly define a graph and the concepts necessary to define some of the existing graph kernels.

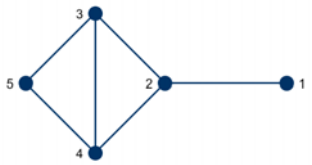


Figure 2. A graph G with adjacency matrix A in Equation 1.

A *finite graph* G consists of a set of vertices $V(G) = \{v_1, v_2, \ldots, v_n\}$ with possible edges between them. The number of vertices in the graph is referred to as the *order* of the graph, in this case n. An edge is denoted by $e = v_i v_j$, meaning the link between vertices v_i and v_j. The set of edges of the graph is denoted by $E(G)$. We consider simple graphs with no loops (an edge from a vertex to itself) or multiple edges between vertices. If two vertices are joined by an edge, we refer to them as being adjacent. If two edges meet at a common vertex, they are also referred to as being *adjacent*. An edge meeting a vertex is *incident* to that vertex.

A *walk* in the graph refers to movement across the graph, starting at a vertex and moving along edges and across other vertices allowing for edges and vertices to be visited more than once. The *length* of the walk is simply the number of edges traversed. A *path* is a walk where no edges or vertices are visited more than once, and a *cycle* is a closed path, i.e. its initial vertex is also its end vertex.

We can describe the graph G by making use of an *adjacency matrix* A, which is an $n \times n$ matrix consisting of zeros and ones, depending on whether or not two vertices are adjacent. We therefore have the ij-th entry given as

$$[A]_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G), \\ 0 & \text{if } v_i v_j \notin E(G), \end{cases}$$

where $1 \leq i, j \leq n$.

Consider the graph G in Figure 2. For simplicity the vertices $v_1, v_2, \ldots, v_5$ have been labeled using only values $1, 2, \ldots, 5$. The adjacency matrix A of G is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{1}$$

The adjacency matrix of a graph contains valuable information, especially if we want to compare two graphs. The sum of the elements in each row or column gives that particular vertex's degree, i.e. its amount of adjacent vertices or neighbours. Also, the powers of the adjacency matrix, say A^4, will give us the number of walks of length 4 in the graph, this can be found for each entry $[A]_{ij}$, that is, from vertex v_i to vertex v_j [6]. This concept is often used in what is called walk-based kernels, together with the idea of a direct product between graphs. To 'walk' on a direct product graph is therefore the same as walking on both graphs simultaneously [7–9]. An example of a direct product between graphs is given in Figure 3.

Graph-with-graph comparisons tend to fall under graph classification problems, while within-graph similarities are more used to cluster graphs.

2.2. Clustering

In terms of clustering or comparing vertices within a graph, we can make use of minimum spanning trees (MST), shared nearest neighbour (SNN), betweenness centrality, highly connected subgraphs, maximal clique enumeration and the kernel k-means, to name but a few.

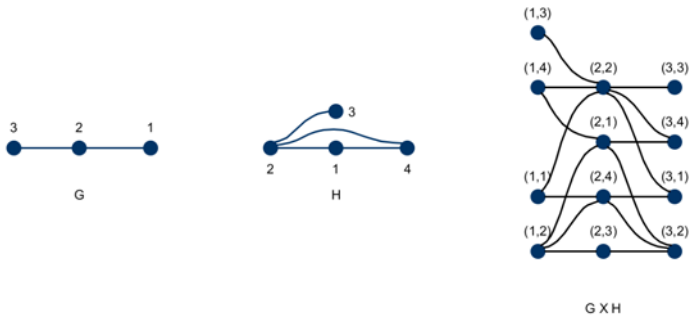


Figure 3. An example of a direct product between two graphs G and H. If the graphs G and H have vertex sets $V(G)$ and $V(H)$, respectively, then the direct product graph of G and H has the vertex set $V(G) \times V(H)$, namely the set of ordered pairs consisting of vertices from G and H. Two vertices (u, u') and (v, v') in $G \times H$ are adjacent if and only if u is adjacent to v in the graph G and u' is adjacent to v' in the graph H.

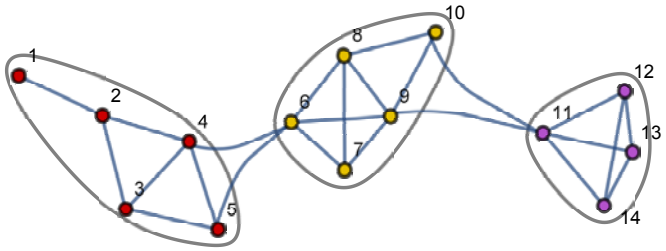


Figure 4. An example of graph clustering done on selected pulsars with respect to their dispersion measures. The lengths of the edges are not taken into consideration since they only represent connections between vertices.

In Figure 4 we have made use of a simple graph clustering based on the dispersion measure of each pulsar. A input threshold μ is specified whereby we form the graph structure. For this specific graph we choose $\mu = 10$ and vertices (pulsars) are adjacent if the difference between their dispersion measures are less than or equal to μ. On this graph we can then choose our method of clustering, depending on the similarities that we want to highlight. We now briefly discuss some of the methods of graph clustering.

A minimum spanning tree can be constructed from any connected weighted graph. A weighted graph is a graph where the edges have been assigned a numerical value which is referred to as its weight. A *tree* is a connected graph that contains no cycles. A *spanning tree* of a graph G is a subgraph of G consisting of all vertices of G. We construct a minimum spanning tree as follows. After choosing any vertex v to begin with, we use Prim's algorithm (see also Kruskal's algorithm) and we construct the tree T by using an incident edge that satisfies

(i) it has the lowest weight,

(ii) it has only 1 incidence with T.

We repeat this process until we have a tree T that contains all the vertices of the original graph, hence spanning. In order to cluster the graph, we start eliminating the maximum weighted edges from T. Since the tree T is a minimum spanning tree, the removal of any edge will result in a disconnected graph, the components of which would form the clusters.

Our aim is to make use of one or more of the existing clustering methods to investigate clustering of pulsars on the $\dot{P}$-P diagram. The main task will be to structure our data effectively

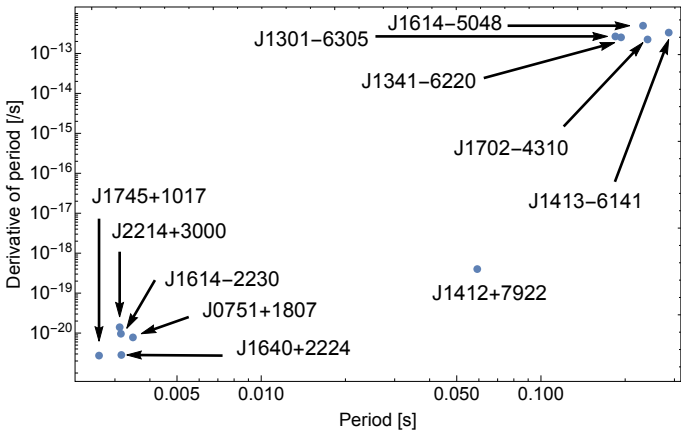


Figure 5. The pulsars in Table 1 plotted against period and period derivative.

in such a manner that the clustering methods could pick up more than just the standard clusters.

3. Application to $\dot{P}$-P diagram

To illustrate one of the most basic clustering methods using graph theory, we shall make use of eleven chosen pulsars and cluster them according to their respective distances on the $\dot{P}$-P diagram. The selection is obviously biased, but sufficient to test the algorithm.

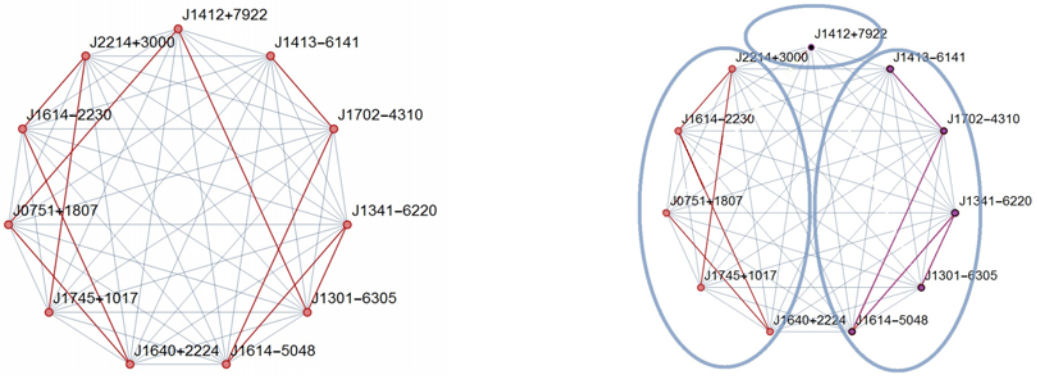
We form a complete graph of 11 vertices, that is, all vertices are adjacent to each other. Next, we assign weights to each edge in the graph according to the physical distances between its incident vertices (pulsars) on the $\dot{P}$-P diagram. We make use of a minimum spanning tree to cluster these pulsars. The pulsars that we have made use of can be found in Table 1. Their positions are plotted in Figure 5.

Now we construct a minimum spanning tree by choosing a vertex and adding edges according to the algorithm mentioned earlier. The spanning tree is shown in Figure 6a (bold edges).

For us to cluster these pulsars according to the spanning tree, we simply remove the heaviest weighted edges from the tree. For every edge we remove, we create a new clustering. If we remove the two heaviest edges from this specific tree, we find the clusters in Figure 6b, corresponding to our $\dot{P}$-P diagram in Figure 5.

Table 1. Pulsars and their $\dot{P}$-P coordinates used in our minimal working example.

PSR	P	Pdot
J2214+3000	0.003119	1.4×10^{-20}
J1614-2230	0.003151	9.62×10^{-21}
J0751+1807	0.003479	7.79×10^{-21}
J1745+1017	0.002632	2.73×10^{-21}
J1640+2224	0.003163	2.83×10^{-21}
J1412+7922	0.059198	4×10^{-19}
J1614-5048	0.231694	4.95×10^{-13}
J1301-6305	0.184528	2.67×10^{-13}
J1341-6220	0.19334	2.53×10^{-13}
J1702-4310	0.240524	2.24×10^{-13}
J1413-6141	0.285625	3.33×10^{-13}



(a) The spanning tree constructed from Table 1. **(b)** The spanning tree clusters formed

Figure 6. Spanning tree construction and clustering of a sample of pulsars

4. Discussion and Conclusions

In this paper we illustrated the usefulness and the adaptability of the graph cluster algorithm within the $\dot{P}$-P diagram by using pulsars with unique characteristics and clustered them accordingly. The MST results are in agreement with the clustering seen in Figure 5. The choice of threshold is problem specific. Optimizing the threshold can be done by several iterations using different threshold values. If the threshold is taken to high, the algorithm will simply allocate a cluster to each data point. For future work we are planning to cluster the entire up-to-date $\dot{P}$-P diagram using a well optimized clustering code.

Acknowledgments

The financial assistance of the South African SKA Project (www.ska.ac.za), and the National Research Foundation of South Africa.

References

[1] Lorimer D R 2005 *Handbook of pulsar astronomy* (Cambridge, UK New York: Cambridge University Press) ISBN 978-0-521-82823-9

[2] Turolla R, Zane S and Watts A L 2015 *Reports on Progress in Physics* **78** 116901 (*Preprint* 1507.02924)

[3] Schaeffer S E 2007 *Computer Science Review* **1** 27 – 64 ISSN 1574-0137

[4] Atwood W *et al.* 2013 *ArXiv e-prints* (*Preprint* 1303.3514)

[5] Campana R *et al.* 2008 *Monthly Notices of the Royal Astronomical Society* **383** 1166–1174 (*Preprint* 0710.3691)

[6] Samatova N F *et al.* 2013 *Practical Graph Mining with R* (Chapman & Hall/CRC) ISBN 143986084X, 9781439860847

[7] Gärtner T, Flach P and Wrobel S 2003 *Conference on learning theory* pp 129–143

[8] Ramon J and Gärtner T 2003 *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences* pp 65–74

[9] Imrich W and Klavzar S 2000 *Product Graphs: Structure and Recognition* (John Wiley and Sons, New York, USA)