Graph-based Methods for Orbit Classification

Abraham Bagherjeiran, Chandrika Kamath *

Abstract

An important step in the quest for low-cost fusion power is the ability to perform and analyze experiments in prototype fusion reactors. One of the tasks in the analysis is the classification of orbits in Poincaré plots generated by the particles in a fusion reactor as they move within the toroidal device. In this paper, we describe the use of graph-based methods to extract features from orbits. These features are then used to classify the orbits into several categories. Our results show that existing machine learning algorithms are successful in classifying orbits with few points, a situation which can arise in data from experiments.

Keywords: orbit, Poincaré plot, classification.

1 Introduction

The quest for low-cost fusion power has led to the construction of devices such as the National Compact Stellarator Experiment (NCSX) at the Princeton Plasma Physics Laboratory (PPPL). These devices allow physicists to perform magnetic confinement experiments which determine the best shape for the hot reacting plasma and the magnetic fields necessary to hold it in place. In addition, advances in computational resources have made possible the computational simulation of these experiments in three dimensions over time. This allows the physicists to design new reactors and select the parameters to be used in experiments. The experimental results are, in turn, used to validate the simulations. Thus, the analysis of data from both simulations and experiments is a key step in the understanding and development of fusion reactors.

Figure 1 shows the schematic of the NCSX. A particle moving around the torus will trace out a three-dimensional trajectory over time. Consider a plane intersecting the torus perpendicular to the magnetic axis–a vertical slice through the torus. Let a point in this plane be the intersection of the trajectory of the particle with the plane as it starts to move through the torus. After it completes one round through the torus, it will likely intersect the plane at a different point. The intersections of this trajectory with the plane form an orbit.

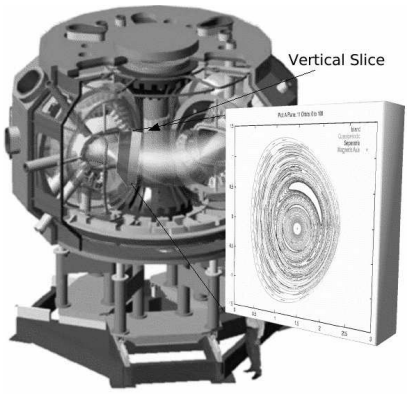


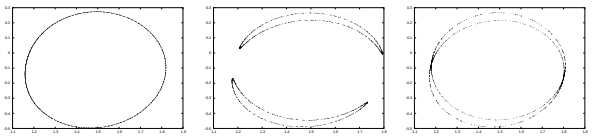
Figure 1: A schematic of the NCSX reactor. Inset shows a plane perpendicular to the magnetic axis illustrating the intersections of the particles.

Depending on the shape of the orbit, it can be assigned a class label. Figure 2 depicts three different orbits: a separatrix, an island chain, and a quasi-periodic orbit. There is also an additional class of stochastic orbits, which we will not consider in the present analysis. Note that the quasi-periodic orbit appears to be a closed curve with no apparent width. The island chain orbit has two distinct islands in this example. The separatrix orbit appears closed but has radial gaps called lobes; there are two such lobes in this orbit. Typically, all the orbits on a plane are provided together in what is referred to as a "puncture plot" or a "Poincaré plot", as in Figure 1, inset.

Orbits from computer simulations usually consist of a thousand or more points. In contrast, plots from experiments consist of 50 to 100 noisy points, which may be too few to correctly identify the shape. In our work, we will consider how the accuracy of a classifier changes as the number of points in an orbit is reduced. If we can correctly classify orbits using a few points, it reduces the time for the extraction of features and leads to a faster turnaround in the analysis.

We next briefly describe our approach to the classification of orbits. Further details are available in [1, 2].

*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA



(a) (b) (c)

Figure 2: Sample orbits from each class. (a) Quasi-periodic, (b) Island Chain, and (c) Separatrix.

2 Orbit Preprocessing

Our approach to orbit classification is derived from a graph-based method described in [6]. First, we compute a graph representation of an orbit in the form of its Euclidean Minimal Spanning Tree (MST). The MST of a graph is a tree that contains all nodes of the graph but the sum of the edge lengths in the tree is minimal. Next, we partition the graph into independent clusters, or subgraphs, by removing edges whose length exceeds a threshold $t_{partition}$. Finally, we assign properties to the nodes and edges of the entire orbit graph as well as all the partitioned subgraphs. We next describe a few illustrative properties extracted from the graphs. A complete list can be found in [1].

- Each node has the *cluster* property, which is the index of the subgraph to which the node belongs.
- The nodes along the diameter of the graph have the property *diameter* set to *true*; otherwise, it is set to $false$. The diameter is defined as the longest shortest path between any two nodes in the graph.
- For each node v, with degree 3 or more, we compute the *branchLength* property (Figure 3) as the length of the longest path starting at v and not passing through any other node along the diameter. Nodes with degree less than 3 are ignored and assigned $branchLength(v) = 0$. Each node is then assigned two Boolean properties. Given thresholds $0 < t_{shallow} < t_{deep}$, $shallowBranch(v)$ is defined as:

$$\begin{array}{cc} true & \text{if } branchLength(v) \geq t_{shallow} \\ false & \text{otherwise} \end{array}$$

and $deepBranch(v)$ is defined as:

$$\begin{array}{cc} true & \text{if } diameter(v) \wedge (branchLength(v) \geq t_{deep}) \\ false & \text{otherwise} \end{array}$$

- The $primaryBranch(v)$ property is defined as:

$$\begin{array}{cc} true & \text{if } diameter(v) \wedge shallowBranch(v) \wedge \\ & (branchLength(v) = \max_w branchLength(w)) \\ false & \text{otherwise} \end{array}$$

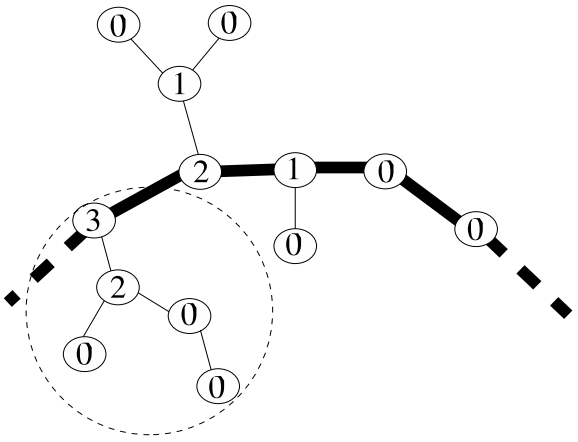


Figure 3: Branch calculation example on a subgraph. The thick edges indicate the diameter of the graph. Node labels indicate the *branchLength* values. Nodes in the circled region are on the *primaryBranch*.

It is assigned to the branch node v whose path is the longest branch path but not necessarily a deep branch. After assigning the *primaryBranch* property to all nodes along the diameter, the value of the property is replicated to all non-diameter descendants of a primary branch node.

3 Features and Classification Algorithms

We next use the properties described in the previous section to create the features for classification algorithms. We consider three approaches: the rule-based approach of [6], which we call the default KAM, a customized version of KAM which has been modified for our data, and several standard machine learning classifiers.

Since orbits can vary in size, all lengths are given as fractions of the diameter length. We use the following notation in the rest of this section.

- f_X: A feature X that is a fraction with range $[0, 1]$.
- n_X: Number of nodes that satisfy property X. Since all nodes have the property $node(v)$, the value n_{node} is the number of nodes in the graph.

3.1 The Default KAM classifier For the default KAM classifier, we first extract the features described in the book [6]. Example features include:

- d: The Euclidean distance between the initial and final nodes of the diameter as a fraction of the total path length between the nodes.
- $n_{shallow}$: Number of shallow branch nodes.

- $f_{shallow}$: Fraction of shallow branch nodes: $n_{shallow}/n_{node}$.

- $n_{clusters}$: The number of clusters found in the graph partitioning process.

- *Origin*: Indicates if the orbit contains the origin of the predefined magnetic axis.

After extracting the features, the KAM classifier applies a set of static rules to determine whether an orbit belongs to one of the classes. For example, an orbit is quasi-periodic if it includes the origin and satisfies the following rule: $(n_{clusters} = 1) \wedge (n_{shallow} < p_{branch}) \wedge (d \leq p_d)$ where p_d is a parameter set by the user. The rules for other orbit classes, the values of the parameters, and the determination of thresholds, are described in [1].

3.2 The Custom KAM classifier Our work on the default KAM approach indicated that we could improve the results by including one additional feature with a new threshold:

- f_{LS}: Fraction of clusters that are line segments. A line segment is a graph whose Euclidean distance between the initial and final node of the diameter is a large fraction of the total diameter path length with respect to the threshold t_{LS}.

Further, to allow more expressive rules in our custom KAM algorithm, we use real-valued instead of Boolean predicates and real-valued combination operators. This allows for a partial (instead of Boolean) evaluation which is tolerant to poorly selected thresholds and parameters. Thus, the quasi-periodic rule evaluation K_Q has three components that mirror the default KAM rules:

$$\begin{aligned} K_{Q_1} &= n_{shallow} \leq p_{branch} \\ K_{Q_2} &= d < p_d \\ K_{Q_3} &= Origin \end{aligned}$$

The final evaluation is the average of the component predicates:

$$K_Q = \frac{1}{3}(K_{Q_1} + K_{Q_2} + K_{Q_3})$$

The remainder of the rules are similarly defined as averages of their respective components predicates. The process for evaluation of the parameters and thresholds is described in [2].

3.3 The Standard Machine Learning Classifiers We use the features extracted for both the default and custom KAM classifiers to train standard machine learning classifiers. We include additional features that convey the underlying statistics of the edges and clusters of the orbits, such as

- μ_{length}: Average edge length.

- σ_{length}: Standard deviation of the edge length.

- u: Diameter uniformity defined as the difference between the average edge length and what would occur if the points were equally spread across the diameter: $\frac{|\mu_{length} - \bar{u}|}{\bar{u}}$. If the points were distributed at equal intervals along the diameter, the average edge length would be $\bar{u} = \frac{1}{n_{diameter}}$.

- μ_d: Average diameter endpoint distance for each cluster.

- $f_{cluster}$: Fraction of clusters to nodes: $\frac{n_{clusters}}{n_{node}}$ or the percent of nodes that are clusters.

These features are used in the following traditional classification algorithms: Naïve Bayes, Multilayer Perceptron, Support Vector Machine, k-Nearest-Neighbor, and Decision Tree, which are available as part of the Weka machine learning toolkit [4].

4 Experimental Results

We perform two experiments to evaluate our approach on 6 plots each containing 100 orbits from simulations of the CDX tokamak at PPPL. The number of points in each orbit is variable. We hand labeled these orbits to create a training set. For instance-based classifiers such as the nearest-neighbor algorithm, we normalized the features to be in the range $[0, 1]$.

Experiment 1: Classification Accuracy We compare the accuracy of various classifiers by performing 10 runs of 10-fold stratified cross-validation. The classifiers are both trained and tested on features extracted from the orbits using the first N points. The number of points for each run is in the set

$$N \in \{50, 100, 200, \ldots, 1000, 1500, 1800, 2000, 2500\}$$

where the maximum number of possible points for any orbit in the dataset was 3000.

Experiment 2: Reduced Number of Points We perform a similar comparison by performing 10 runs of 10-fold stratified cross-validation. The classifiers are trained on features from orbits with 2500 points, but tested on orbits with N points. This allows us to determine the accuracy when we train on orbits with a large number of points and test on orbits with a reduced

number of points. The classifiers used the same orbits in the same order as in the previous comparison; only the number of points was changed.

4.1 Results and Discussion Figure 4 shows the time to extract features for an average orbit as the number of points is increased. This indicates the trade-offs between the higher accuracy resulting from more points and the corresponding increase in the cost of feature extraction.

Figure 5 shows the results from the two experiments, indicating the average accuracy. The results for Experiment 1 show that the customized KAM classifier is a vast improvement over the original KAM in most cases. Also, the features allowed the other traditional classification algorithms to outperform both the KAM classifiers in most cases. Interestingly, the simplest classifiers–decision tree and nearest neighbor–appear to be the overall winners in this comparison.

The results for Experiment 2, however, show that training on a large number of points and then testing on a small number of points does not seem to improve the accuracy results, significantly. Comparing the results of the two experiments, we observe that for orbits with more than 1500 points, training on orbits with a large number of points does give similar accuracy to that of Experiment 1. However, for orbits with a smaller number of points (50 to 400), the accuracy is significantly improved when both training and testing on these low-point orbits as in Experiment 1.

This effect of a reduced number of points in an orbit is best explained in Figure 6. The figure shows 3 example orbits with 150 and 1000 points. At 1000 points, the MST correctly identifies the outline of the shapes. At 150 points, however, the separatrix contains far too few branches because the MST forms a "zig-zag" pattern connecting points that are branches at 1000 points. In the quasi-periodic orbit, KAM detects multiple clusters misclassifying both it and the separatrix as island chains. At 1000 points, only the island chain orbit has several clusters. Although the feature-based classifiers perform better than KAM, they use the features generated by the graph-based approach of KAM. The results, therefore, reflect the quality of these features.

The reliance on graph-based KAM features also explains why the feature-based algorithms perform better when trained and tested on the same number of points. Orbits with few points clearly appear different from orbits with many points. As a result, the features extracted are different. Therefore, one cannot expect a feature-based classifier trained on orbits with many points to perform well on orbits with few points.

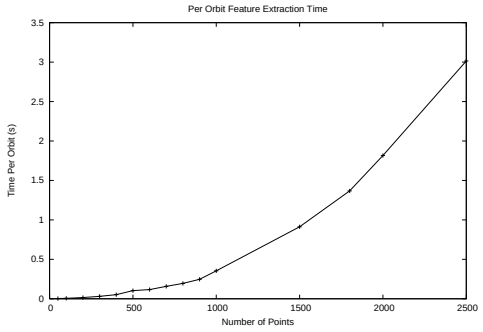


Figure 4: Per-orbit feature extraction time for orbits with an increasing number of points.

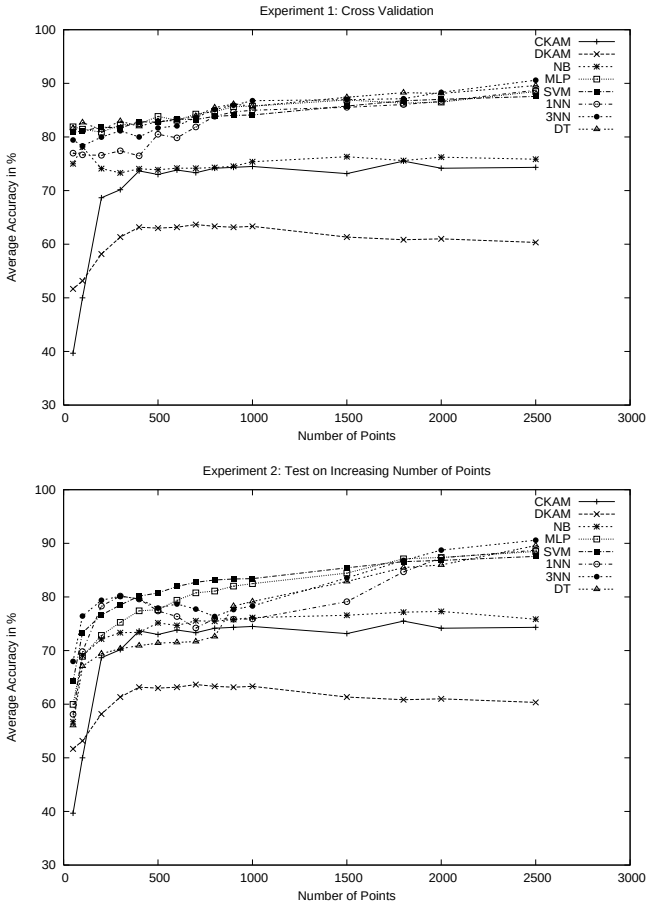


Figure 5: Comparative performance for Experiment 1 on accuracy of classifiers (top) and Experiment 2 on the effect of reduced number of points (bottom).

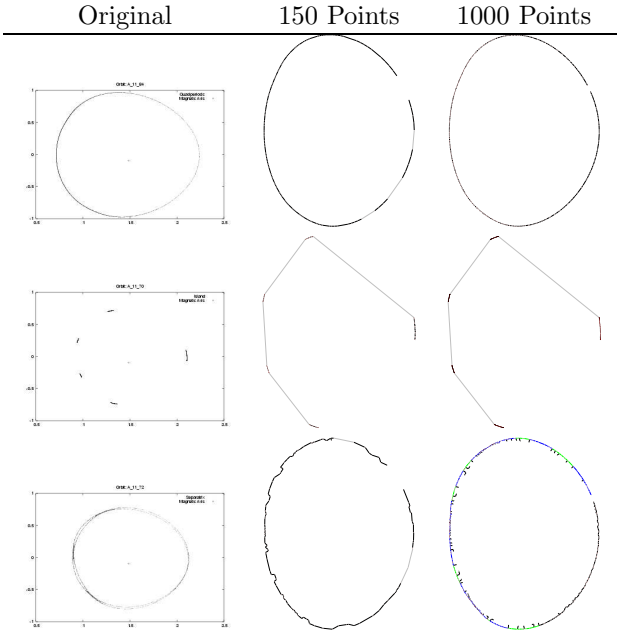


Figure 6: Orbits illustrating the effects of reduced number of points for the quasiperiodic (top), island chain (middle), and separatrix orbits (bottom).

5 Related Work

The analysis of Poincaré plots, in particular, the problem of classification of orbits, has been studied in both the physics and machine learning communities. When the points that make up an orbit are available, the approach typically used is to classify the orbits based on a description extracted from the points. In spatial aggregation [7, 5], an approach based on visual reasoning starts with an image-like input (e.g. the points of an orbit), and transforms the point representation into more economical symbolic representations using techniques similar to those in computer vision. Such techniques are also used to manipulate topological structures in spatial data. The early work in the use of geometric and spatial reasoning includes the analysis of Hamiltonian systems in the KAM approach [6] used in this paper and the MAPS system, which designs control laws based on a geometric analysis of the state equations of a dynamical system [7].

More recently, the area of Qualitative Spatial Reasoning [3] has evolved to provide a solution to problems where numerical methods cannot describe the geometric and topological structures in data sets required to answer high-level spatial queries. This idea of spatial reasoning is broadly applicable to problems in scientific domains such as geographic information systems, meteorological and fluid flow analysis, and CAD systems.

6 Conclusion

In this paper, we considered the problem of classifying orbits from fusion devices. We showed how an existing rule-based algorithm called KAM, which uses features extracted from the graph representation of the orbit, can be customized to be more suitable for our dataset. Comparing the performance of several feature-based classification algorithms, we find that they outperform both the default and our customized KAM classifiers, have high accuracy, and good tolerance for orbits with few points. They are also sufficiently straightforward to implement in the analysis pipeline after off-line training.

Our future work will concentrate on evaluation of more sophisticated single-orbit features as well as further testing with orbit data from different simulations.

Acknowledgments UCRL-CONF-215802: This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

We gratefully acknowledge the domain expertise provided by Neil Pomphrey and Don Monticello at PPPL. We also acknowledge Scott Klasky for introducing us to this problem and for his support of this work.

Abraham Bagherjeiran is a graduate student at the University of Houston. This work was done when he was a student intern at LLNL during Summer 2005.

References

[1] A. Bagherjeiran and C. Kamath. Graph-based methods for orbit classification (extended version). Technical Report UCRL-CONF-215802, Lawrence Livermore National Laboratory, 2005.

[2] A. Bagherjeiran and C. Kamath. Graph-based techniques for orbit classification: Early results. Technical Report UCRL-TR-215690, Lawrence Livermore National Laboratory, 2005.

[3] C. Bailey-Kellogg and F. Zhao. Qualitative spatial reasoning: extracting and reasoning with spatial aggregates. *AI Magazine*, 24(4):47–60, 2004.

[4] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[5] K. Yip and F. Zhao. Spatial aggregation: Theory and applications. *Journal of Artificial Intelligence Research*, 5:1–26, 1996.

[6] Kenneth Man-Kam Yip. *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer*. MIT Press, 1991.

[7] F. Zhao. Extracting and representing qualitative behaviors of complex systems in phase space. *Artificial Intelligence*, 69:51–92, 1994.