

5-2016

## Computing All Isolated Invariant Sets at a Finite Resolution

Martin Salgado-Flores  
*College of William and Mary*

Follow this and additional works at: <https://scholarworks.wm.edu/honorsthesis>



Part of the [Dynamic Systems Commons](#), and the [Non-linear Dynamics Commons](#)

---

### Recommended Citation

Salgado-Flores, Martin, "Computing All Isolated Invariant Sets at a Finite Resolution" (2016).  
*Undergraduate Honors Theses*. Paper 971.  
<https://scholarworks.wm.edu/honorsthesis/971>

This Honors Thesis is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Undergraduate Honors Theses by an authorized administrator of W&M ScholarWorks. For more information, please contact [scholarworks@wm.edu](mailto:scholarworks@wm.edu).

# Computing All Isolated Invariant Sets at a Finite Resolution

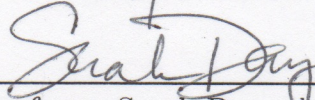
A thesis submitted in partial fulfillment of the requirement  
for the degree of Bachelor of Science in Mathematics from  
The College of William and Mary

by

Martin Salgado-Flores

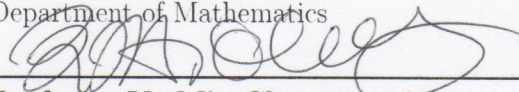
Accepted for

*Honors*



Professor Sarah Day, advisor

Department of Mathematics



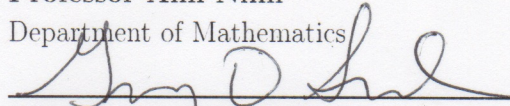
Professor Yu-Min Chung, co-advisor

Department of Mathematics



Professor Anh Ninh

Department of Mathematics



Professor Greg Smith

Department of Applied Science

Williamsburg, VA

May 3, 2016

## Acknowledgements

A very gracious thank you to Professor Sarah Day and Professor Yu-Min Chung for advising and guiding me throughout this project. Also, I would like to thank Dr. William Kalies for all his help with CDS.

## Abstract

Conley Index theory has inspired the development of rigorous computational methods to study dynamics. These methods construct *outer approximations*, combinatorial representations of the system, which allow us to represent the system as a combination of two graphs over a common vertex set. *Invariant sets* are sets of vertices and edges on the resulting digraph. Conley Index theory relies on *isolated invariant sets*, which are maximal invariant sets that meet an isolation condition, to describe the dynamics of the system. In this work, we present a computationally efficient and rigorous algorithm for computing all isolated invariant sets given an outer approximation. We improve upon an existing algorithm that “grows” isolated invariant sets individually and requires an input size of  $2^n$ , where  $n$  is the number of grid elements used for the outer approximation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Background</b>	<b>4</b>
<b>3</b>	<b>Algorithms</b>	<b>11</b>
3.1	ALLIIS Algorithm . . . . .	11
3.2	ALLIIS with LookUp Algorithm . . . . .	13
<b>4</b>	<b>Results and Discussion</b>	<b>16</b>
4.1	Application of Algorithms on $\mathcal{F}_{selfloop}$ . . . . .	16
4.2	Application of Algorithms on $\mathcal{F}_{complete}$ . . . . .	17
4.3	Application of Algorithms on $\mathcal{F}_{tent}$ . . . . .	18
4.4	Discussion . . . . .	21

# Chapter 1

## Introduction

Dynamical systems theory seeks to understand the mathematics of motion. As such, dynamical systems theory provides a mathematical way to study a variety of natural phenomena, playing an important role in understanding physical, biological, social, and economic systems. Mathematicians, including Henri Poincaré, George Birkhoff, Stephen Smale, and Alexander Sharkovsky, made significant contributions to dynamical systems theory throughout the 19th and 20th century. The work of these mathematicians established new methods of understanding dynamical systems, both globally and locally. Since then, old notions of the nature of motion were shattered by surprising discoveries, such as the discovery of chaotic dynamics. This new mathematics has since been applied to the natural world. They have had far-reaching, deep, and meaningful impacts on the natural sciences.

Within dynamical systems, two modeling approaches to studying systems include continuous-time dynamical systems, i.e. differential equations, and discrete-time dynamical systems, i.e. iterated maps. The work discussed in this paper will focus on the latter. Iteration of a map  $f : X \rightarrow X$ , on the phase/state space  $X$  defines a discrete-time dynamical system. The forward iterates of the map are,

$$x_{n+1} := f(x_n), n \in \mathbb{N}, x_0 \in X$$

and  $\gamma_{x_0}^+ = x_0, x_1, \dots$  is the *forward trajectory* of  $x_0$ . We can also define a bi-infinite sequence,

$$x_{n+1} := f(x_n), n \in \mathbb{Z}, x_i \in X$$

yielding a *full trajectory*  $\gamma_{x_0} = \dots, x_{-1}, x_0, x_1, \dots$  through  $x_0$ .

A forward trajectory describes the deterministic path starting at a point in the state space as the map is applied to that point. Despite the determinism of these maps, the behavior of dynamics may be surprising. A major result from the study of dynamical systems is the phenomena of chaos (discussed further in [1]). One

example discussed in this paper is the famous tent map, which has been shown to be chaotic under the parameter values we choose to study here, also discussed in [1].

In the mid-20th century, a mathematician named Charles Conley developed a topological theory based on Morse theory that facilitated a rigorous computational approach for extracting the dynamics of systems. Following the work of Conley, a group of mathematicians has since been developing computational tools that allow further study of dynamical systems. To this end, software has been developed that uses the results of the theory to prove the dynamics of maps. The work presented in this paper attempts to improve upon existing methods of the software presented in [3] and [4].

Important objects of study are invariant sets when computing dynamics. These are sets  $S \subset X$ , such that  $f(S) = S$ . An example of an invariant set is a fixed point, i.e. a point  $x \in X$  such that  $f(x) = x$ . If  $f^k(x) = x$  for some  $k$ , the set  $\{x, f(x), f^2(x), \dots, f^{k-1}(x)\}$  is a periodic orbit, another invariant set. These sets are important in the dynamics, since bounded trajectories limit to invariant sets. Isolated invariant sets are invariant sets that meet an isolation condition (discussed in Chapter 2). They are the basis for computing a Conley index, which is then used to prove the dynamics of a map. The Conley index is designed to extract information about the existence and structure of isolated invariant sets. We present an algorithm designed to find all combinatorial isolated invariant sets, that is, those that are representable in our chosen computational framework. We seek to determine a reasonably efficient method that conducts such a search. Lastly, this paper will attempt to provide possible directions for further improvement in achieving this goal.

# Chapter 2

## Background

Understanding the dynamics of maps is a central goal of dynamical systems theory. Using trajectories, it is possible to study the long term behavior of the system. Bounded trajectories limit to invariant sets, hence we focus on finding and understanding invariant sets including, fixed points, periodic orbits, etc. However, we come across two major problems when analyzing maps by their trajectories. First, there can be infinitely many distinct trajectories generated by a map. Tracking each trajectory is infeasible. A second problem stems from computing the trajectories. Often trajectories will pass through values that are unrepresentable by a computer. These values are rounded off by the computer. This becomes especially problematic when studying chaotic orbits, since the orbits may behave drastically differently under slight differences in initial conditions (see [1] for definition of chaotic orbits).

Instead, we need to find a way to make our problem finite. In this spirit, methods have been developed that create a uniform cubical grid, defined in Section 2.1 below in Definition 2.0.1, over a rectangular region in  $X$  and use Conley Index theory to compute dynamical information about the map (see [2]). In what follows, we assume the  $X$  is a rectangular region, perhaps by restricting it to a rectangular domain of interest.

**Definition 2.0.1.** *A uniform cubical grid  $G$  at depth  $d$  produced by a subdivision of a rectangular set  $X = \prod_{k=1}^n [x_k^-, x_k^+] \subset \mathbb{R}^n$  is defined as  $\{\prod_{k=1}^n [x_k^- + \frac{i_k r_k}{2^d}, x_k^+ + \frac{(i_k+1)r_k}{2^d}] \mid i_k \in \{0, \dots, 2^d - 1\}\}$  where  $r_k = x_k^+ - x_k^-$  is the radius of  $X$  in the  $k$ th coordinate and the depth  $d$  is a nonnegative integer.*

The grid subdivides the space into (closed) boxes, i.e. the product of closed intervals whose interiors are not overlapping. After placing a grid over our space, we can construct a combinatorial/finite representation of  $f$ . With this representation, we will have made our problem finite and computable. The following definitions



provide us with the construction of graphs that form a *combinatorial representation* of the dynamical system.

**Definition 2.0.2.** Let  $V$  to be the vertex set consisting of boxes in the grid  $G$ , that is for each box  $b \in G$  there is a vertex  $v \in V$ , such that  $v = b$ .

**Definition 2.0.3.** Given the vertex set  $V$  for the grid  $G$  on  $X$ . Define the undirected edge set  $E_X$  to be

$$E_X = \{(B_1, B_2) \in G \times G \mid B_1 \cap B_2 \neq \emptyset\} \subset V \times V$$

**Definition 2.0.4.** For  $f : X \rightarrow X$  and the vertex set  $V$  for the grid  $G$  on  $X$ . Define the directed edge set  $E_f$  to be

$$E_f = \{(B_1, B_2) \in G \times G \mid x \in B_1, f(x) \in B_2\} \subset V \times V$$

The undirected edges of  $E_X$  represent the adjacency information for boxes in the phase space and the directed edges of  $E_f$  give possible transitions between boxes under the map  $f$ . Thus, with this information, we can create the following definition.

**Definition 2.0.5.** Given  $f : X \rightarrow X$  and  $G$  a grid on  $X$ . The triple,  $\mathcal{F} = (V, E_X, E_f)$ , is a *combinatorial representation* of the dynamical system. This encodes a (minimal) outer approximation of the map  $f : X \rightarrow X$  in that it contains the smallest outer bounds on images under  $f$ .

The combinatorial representation,  $\mathcal{F}$ , contains information about phase space and the action of the map  $f$ .  $\mathcal{F}$  offers a coarse picture of the dynamical system. As we increase resolution, i.e. subdivide the grid further, the dynamics of the combinatorial representation may resemble many aspects of the dynamics of the underlying map. At coarse resolutions (few grid elements), the combinatorial representation envelopes and represents many possible maps, typically containing little information about the dynamics of the target map. For example, consider the complete graph, i.e. a directed graph where every element is connected to every other element in the graph. Any underlying map  $f : X \rightarrow X$  may have the complete graph as its combinatorial representation if there are too few boxes that cover the map. This is considered a “low resolution”. For any isolated invariant set in  $f$ , however, there is an appropriate resolution that allows us to represent this set in a combinatorial representation.

To illustrate this approach we consider the tent map  $f : [0, 1] \rightarrow [0, 1]$  defined by,

$$f(x) = \begin{cases} rx & : \text{if } x \leq 0.5 \\ r(1 - x) & : \text{if } x > 0.5 \end{cases}$$

The tent map is a common conceptual model when studying the dynamics of iterative maps. Despite its simple formulation, the tent map gives rise to surprisingly complicated dynamics. At certain parameters, the tent map has been shown to be chaotic and topologically equivalent, via a change of coordinates, to the famous logistic map (see [1]). In Figure 2.1, we have a cobweb diagram of a trajectory from the tent map. This cobweb diagram is read by taking an initial point  $x$  in the domain and tracing a vertical line to the value  $f(x)$  under the map. Then, the horizontal line from  $f(x)$  to the diagonal  $y = x$  places  $f(x)$  in the domain. Iteration of this process produces a trace in the graph of a trajectory from an initial point  $x$ . In Figure 2.1, the trajectory is shown to eventually become periodic, since it eventually repeats. However, in Figure 2.2, we can see a seemingly more chaotic trajectory from a different initial condition.

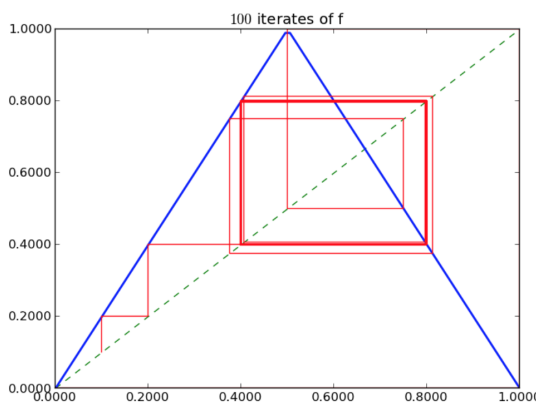


Figure 2.1: Example of a "periodic" trajectory on the tent map with parameter  $r = 2$ .

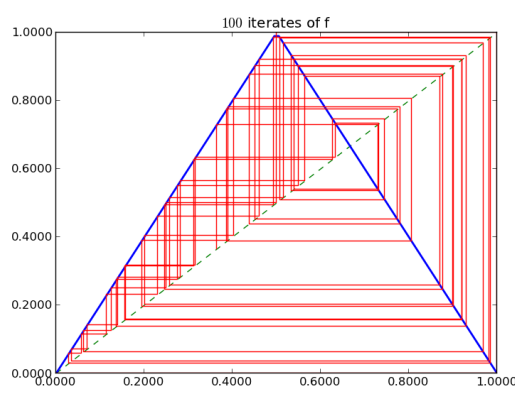


Figure 2.2: Example of a "chaotic" trajectory on the tent map with parameter  $r = 2$ .

Studying the eventual behavior of the tent map can be computationally difficult. The chaotic nature of the map implies that a small error in initial conditions used to start a trajectory can lead to vastly different behavior. Instead, to study the tent map, we can create a combinatorial representation,  $\mathcal{F}_{tent}$ . In Figure 2.3,

we place a grid over the interval  $[0, 1]$  to form the vertex set

$$V = \{[0, 0.25], [0.25, 0.5], [0.5, 0.75], [0.75, 1]\}. \quad (2.1)$$

To simplify notation, we label the intervals in  $V$  as 0, 1, 2, 3 respectively. So,

$$V = \{0, 1, 2, 3\}. \quad (2.2)$$

Now, we can create

$$E_X = \{(0, 0), (0, 1), (1, 1), (1, 0), (1, 2), (2, 2), (2, 1), (2, 3), (3, 3), (3, 2)\}. \quad (2.3)$$

Figure 2.5 shows the graphical representation of  $E_X$ . In Figure 2.4, we can see how to construct  $E_f$ . The shaded regions in Figure 2.4 represent the range for each domain interval in the grid. Now, we can create

$$E_f = \{(0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 0), (3, 1), (3, 2)\}. \quad (2.4)$$

Figure 2.6 shows the graphical representation of  $E_f$ . With  $V$ ,  $E_X$ , and  $E_f$ , we have now constructed a combinatorial representation,  $\mathcal{F}$  of the tent map. This combinatorial representation is very coarse with only a depth of 2, or resolution of  $2^2 = 4$  elements. However, if we were to increase our resolution, we would obtain more information about the dynamics of the tent map.

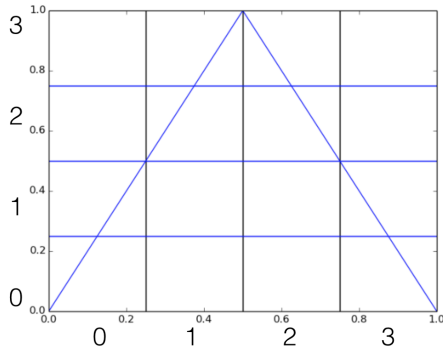


Figure 2.3:  $E_X$  of Tent Map (2.3). The space is divided into closed intervals, forming a cubical grid  $G$ , with  $|G| = 4$ .

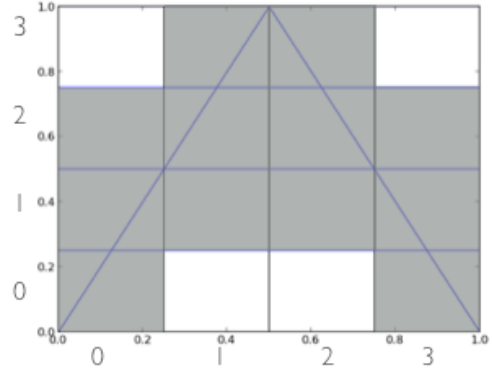


Figure 2.4:  $E_f$  of Tent Map (2.4). The shaded region represents minimal outer bounds on the image of each interval in  $G$ , with  $|G| = 4$ .

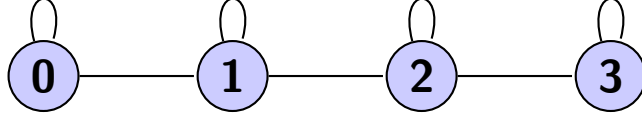


Figure 2.5: The Undirected Graph  $(V, E_X)$  from (2.3) for the Tent Map at  $|G| = 4$ .

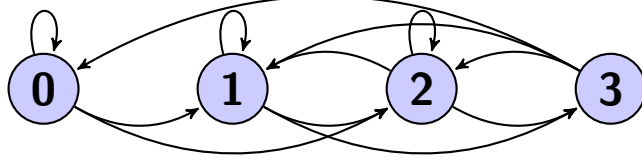


Figure 2.6: Directed Graph of  $(V, E_f)$  from (2.4) for the Tent Map at  $|G| = 4$

Towards our goal of using  $\mathcal{F}$  to study invariant sets, we define the *combinatorial invariant sets* of a region in terms of  $E_f$  and  $E_X$  from the following definitions.

**Definition 2.0.6.** A *combinatorial (full) trajectory* for  $u_0 \in V$  is

$$\gamma_{u_0} = (u_n)_{n \in \mathbb{Z}} \text{ where } (u_n, u_{n+1}) \in E_f.$$

**Definition 2.0.7.** A *combinatorial invariant set* of a set  $S \subset V$  is

$$\text{Inv}(S) = \{u \in S \mid \exists \text{ a combinatorial trajectory } \gamma_u \subseteq S\}.$$

There are many invariant sets in  $\mathcal{F}$  that do not give us meaningful information about the dynamics. For example in a complete graph, every vertex subset  $V' \subseteq V$  is an invariant set. Thus, we restrict our attention to combinatorial invariant sets that satisfy an isolation condition. In other words, restricting our attention to the combinatorial invariant sets that give us the meaningful information about dynamic structure of our system. These *isolated invariant sets* are used to compute a Conley index, which in turn is used to prove results about the actual dynamics of the underlying system (see [2] for a further discussion). Although not the focus in this discussion, sample results obtained using Conley indices in this manner including proving fixed points, periodic points, and chaotic orbits is found in [4]. But first, we must define a neighborhood of a set in  $V$ ,

**Definition 2.0.8.** A *combinatorial neighborhood* of  $S$  is defined as  $o(S) := \{v \in V \mid (u, v) \in E_X \text{ for some } u \in S\}$ .

**Definition 2.0.9.** A set  $\mathcal{S} \subseteq V$  is an *isolated invariant set* if  $\mathcal{S} = \text{Inv}(o(\mathcal{S}))$ . Equivalently, if  $S = \text{Inv}(o(S))$ , we say that  $S$  *satisfies the isolation condition*.

Before computing these isolated invariant sets, we should establish some intuition regarding the resolution of our combinatorial representation. The relation between resolution of the combinatorial representation of a dynamical system and the underlying dynamics of the system is important. We can ask how fixed points, periodic points, etc, of the map are stored in a combinatorial representation. We can quickly see that a fixed point should appear as a selfloop in  $E_f$ . That is, for a fixed point  $x \in u \subseteq V$ , we have  $(u, u) \in E_f$ . Similarly, periodic orbits appear as cycles. At coarse resolutions we usually obtain a representation that is close to the complete graph, since we have fewer elements over the same space, resulting in a more interconnected graph. We introduce  $\mathcal{F}_{complete}$  as one example. For illustration, let  $V = \{0, 1, 2, 3\}$  as in the tent map example in Figure 2.3. Let  $E_X$  to be the set in (2.3), we define  $E_f = V \times V$ . Then,  $\mathcal{F}_{complete} = (V, E_X, E_f)$ .

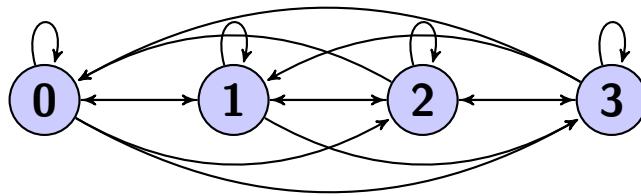


Figure 2.7: Directed Graph  $(V, E_{complete})$  for  $\mathcal{F}_{complete}$

As a second example, consider the graph depicted in Figure 2.9. This is a combinatorial representation for the map,  $f : [0, 1] \rightarrow [0, 1]$ ,  $f(x) = 0.3$  on the same grid used for the previous examples. For this map,  $x = 0.3$  is a superstable fixed point, that is all points in  $[0, 1]$  are immediately mapped to the fixed point after one iteration of  $f$ .

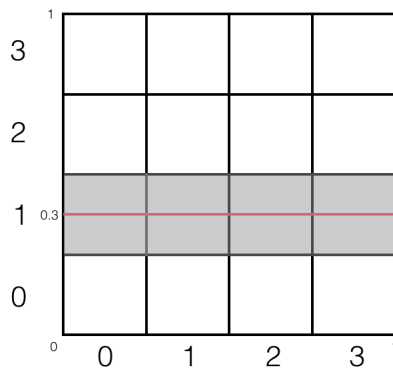


Figure 2.8: The combinatorial representation of  $f(x) = 0.3$ .

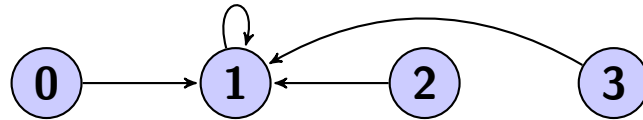


Figure 2.9: Directed Graph Representation of the Single Selfloop example

Note that  $E_f = \{(v, u) \mid u, v \in V \text{ and } 0.3 \in u\}$ . So, there is a selfloop at  $u$  and the in-degree for  $u$  (the number of edges into  $u$ ) in  $E_f$  is  $|V|$ , the number of elements in the grid. As we increase resolution, the vertex set grows, but there is always only one selfloop  $(u, u)$ .

# Chapter 3

## Algorithms

The motivation to find all isolated invariant sets of a dynamical system stems from the importance of these sets in describing the eventual behavior of the system. However, computing all isolated invariant sets is no trivial task. As discussed in the previous section, we must first create a combinatorial representation of the dynamical system that a computer can process. Now, we must search for isolated invariant sets in the combinatorial representation that we have created. A naive approach is to examine all subsets of the vertex set in the combinatorial representation, testing each to determine whether it is both invariant and isolated. This forms the *power set*, the collection of all subsets of  $V$ . For example the *power set* of a set  $S = \{0, 1\}$  is  $P(S) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$ . Note that the size of the power set is  $|P(S)| = 2^{|S|}$ . So, for  $V$ , the power set increases as  $2^{|V|}$ . Improving the resolution of  $\mathcal{F}$  by increasing the size of  $V$  leads to an exponential increase in the size of  $P(V)$ . Since we need high resolutions to prove certain dynamics of the map, the power set method becomes unreasonably large and time consuming. Thus, there is a need for an algorithm that significantly reduces this search. The following algorithm is one such attempt at a solution.

### 3.1 ALLIIS Algorithm

The algorithm begins with the full grid,  $V$ , and stores its maximal invariant set in the a list sets that will be processed, denoted TBP. Then, we remove a set in TBP of maximal length (Step 1). We test if this set is isolated and, if it is isolated, we store it in ALLIIS\_list (Step 2). Next, we look at the maximal invariant sets of subsets of one element less (Step 3). If these subsets are not already in TBP, either as listed sets are as subsets of listed sets, then they are placed in TBP. Next, we process the next set of maximal length in TBP the same way as the full

graph was processed (Step 4). By continuing this process, we guarantee that all isolated invariant sets are found, shown in Theorem 3.1.4.

**Algorithm: ALLIIS**

Begin with  $TBP = \{\text{Inv}(V)\}$ ,

1. Remove a set of maximal length from TBP, call this  $V'$ .
2. If  $V' = \text{Inv}(o(V'))$ , then place  $V'$  in ALLIIS\_list.
3. Now,  $TBP = TBP \cup \{\text{Inv}(V' - \{v\}) \mid v \in V'\}$ .
4. Repeat steps 1-3 until  $TBP = \emptyset$ .

Output: ALLIIS\_list

The following is the justification for the algorithm,

**Lemma 3.1.1.** The ALLIIS algorithm will terminate.

*Proof.* It suffices to show that eventually  $TBP = \emptyset$ . Initially,  $TBP = \{\text{Inv}(V)\}$ . At each step a set of maximal length in TBP is removed. While processing this set, every set that is placed in TBP is of a strictly smaller size. For any given size, there are a finite number of sets of that size. Once all sets of a size are processed, the maximal length of sets in TBP decreases. Thus, eventually, the sets in TBP are of size 0. In other words,  $TBP = \emptyset$  and the ALLIIS algorithm terminates.  $\square$

**Lemma 3.1.2.** If a set  $S \subseteq V$  and  $S = \text{Inv}(o(S))$ , then  $S \subseteq \text{Inv}(V)$ .

*Proof.* For contradiction, suppose  $S \not\subseteq \text{Inv}(V)$ . Now consider,  $M = S \cup \text{Inv}(V)$ .  $M$  is invariant. Also,  $\text{Inv}(V) \subset M \subseteq V$ . Then,  $\text{Inv}(V)$  is not the maximal invariant set of  $V$ , which is a contradiction.  $\square$

**Corollary 3.1.3.** Let  $S = \text{Inv}(o(S))$ . If  $S \subsetneq S'$ , then  $S \subseteq \text{Inv}(S' - \{i\})$  for some  $i \in S'$ .

*Proof.* For  $S \subsetneq S'$ ,  $S \subseteq S' - \{i\}$  for some  $i \in S'$ . By the arguments used to prove Lemma 3.1.2,  $S \subseteq \text{Inv}(S' - \{i\})$ .  $\square$

**Theorem 3.1.4.** If TBP is initialized as the set  $\{\text{Inv}(V)\}$ , then ALLIIS\_list will contain all the isolated invariant sets in  $V$  by the termination of ALLIIS.

*Proof.* Note that for all isolated invariant sets  $S \subseteq V$ ,  $S \subseteq \text{Inv}(V)$ .

At Step 1,  $V' = \text{Inv}(V)$ , if  $V' = \text{Inv}(o(V'))$ , then place  $V'$  in ALLIIS\_list.

Now, for  $V' \in TBP$ , suppose for some isolated invariant set  $S \subseteq V'$ . Let  $V''$  be some child of  $V'$ , that is  $V'' = \text{Inv}(V' - \{v\})$  for some  $v \in V'$ .



Claim:  $S$  is not lost.

Case 1:  $S = V'$ , then  $S$  is added to TBP in Step 2.

Case 2:  $S \subsetneq V'$ , then  $S \subseteq V''$  by Corollary 3.1.3.  $V''$  is added to TBP in Step 3.

In other words,  $S$  is not lost. Since, TBP terminates by Lemma 3.1.1, we have that ALLIIS\_list contains all isolated invariant sets in  $V$ .  $\square$

## 3.2 ALLIIS with LookUp Algorithm

An obvious drawback to the ALLIIS algorithm is that given  $\mathcal{F}$  with a well-connected graph  $(V, E_f)$ , the algorithm will search through a set near the size of the power set. For example  $\mathcal{F}_{complete}$  from Chapter 2 will search the power set since all subsets are invariant. However, we can avoid this problem. Let us suppose for nonempty sets  $S = \text{Inv}(S)$ , we can compute the smallest isolated invariant set  $\text{IIS}(S)$  that contains  $S$  ([4] contains an algorithm called `grow_isolated` for producing  $\text{IIS}(S)$ ). We can further reduce the number of sets that must be added to TBP. This reduction is based on the following lemma and corollary.

**Lemma 3.2.1.** If  $S_i \subseteq V' \subseteq V$  but  $\text{IIS}(S_i) \not\subseteq V'$ , then for any isolated invariant set  $S \subseteq V'$ ,  $S_i \not\subseteq S$ .

*Proof.* Note that  $S_i = \text{Inv}(S_i)$ . Suppose  $S_i \subseteq V' = \text{Inv}(V')$  and  $\text{IIS}(S_i) \not\subseteq V'$ . So, for any  $S \subseteq V'$ ,  $\text{IIS}(S_i) \not\subseteq S$ . Otherwise, if  $\text{IIS}(S_i) \subseteq S$ , then  $\text{IIS}(S_i) \subseteq V'$ , a contradiction. Thus,  $S_i \subseteq \text{IIS}(S_i) \not\subseteq S$ . So,  $S_i \not\subseteq S$ .  $\square$

We can conclude from this lemma that if we process a set  $V'$  such that  $S_i \subset V'$ , but  $\text{IIS}(S_i) \not\subseteq V'$ , then for any isolated invariant set  $S \in V'$ ,  $S_i \not\subseteq S$ . So, we must cut at least one element of  $S_i$  from  $V$  in order to find an isolated invariant set in  $V$ .

**Corollary 3.2.2** (Directed cut). Suppose  $S_i \subseteq V' \subseteq V$  but  $\text{IIS}(S_i) \not\subseteq V'$ . For any isolated invariant set  $S \subsetneq V'$ , then  $S \subseteq \text{Inv}(V' - \{v\})$  for some  $v \in S_i$ .

*Proof.* From Lemma 3.2.1, we have  $S_i \not\subseteq S$ . So,  $S \subseteq V' - \{v\}$  for some  $v \in S_i \subseteq V'$ . By Corollary 3.1.3,  $S \subseteq \text{Inv}(V' - \{v\})$  for some  $v \in S_i$ .  $\square$

Following proof of Corollary 3.2.2, note that Corollary 3.2.2 is stated in a form that allows for a direct replacement of Step 3 in ALLIIS with the directed cut set  $S_i$  listed in Step 3 of ALLIIS with Look Up below. We reduce the number of new sets produced in Step 3 from at most  $|V|$  in  $\{\text{Inv}(V' - \{v\}) \mid v \in V\}$  to at most

$|S_i| < |V|$  in  $\{\text{Inv}(V' - \{v\}) \mid v \in S_i\}$ , In order to take advantage of what could be a drastic reduction in the number of sets produced in Step 3, we populate a look up table with pairs  $(S_i, \text{IIS}(S_i))$  with  $S_i$  very small. The reason for choosing small recurrent sets to fill the look-up table is that the subsets of  $\text{IIS}(S_i)$  are not isolated. So,  $\text{IIS}(S_i)$  are the smallest isolated invariant sets in  $V$ . The following is a sketch of the look-up table, where each  $S_i$  is a recurrent set. This table is ordered from smallest to largest length of  $S_i$ . The right column is constructed as the smallest isolated invariant set containing  $S_i$ .

**LookUpTable**

$S_i$	Isolated invariant set ( $\text{IIS}(S_i)$ )
$S_1$	$\text{IIS}(S_1)$
$\vdots$	$\vdots$
$S_n$	$\text{IIS}(S_n)$

The following is the improved algorithm with the look-up table,

**Algorithm: ALLIIS with LookUp**

Begin with  $\text{TBP} = \{\text{Inv}(V)\}$

1. Remove a set of maximal length from TBP, call this  $V'$ .
2. If  $V' = \text{Inv}(o(V'))$ , then place  $V'$  in ALLIIS\_list.
3. For the first  $S_i \in \text{LookUpTable}$  such that  $S_i \subset V'$  and  $\text{IIS}(S_i) \not\subset V'$ , set  $\text{TBP} = \text{TBP} \cup \{\text{Inv}(V' - \{v\}) \mid v \in S_i\}$ .  
If no such  $S_i$  exists set  $\text{TBP} = \text{TBP} \cup \{\text{Inv}(V' - \{v\}) \mid v \in V'\}$ .
4. Repeat steps 1-3 until  $\text{TBP} = \emptyset$ .

Output: ALLIIS\_list

**Lemma 3.2.3.** The ALLIIS with Look-Up algorithm will terminate.

*Proof.* The argument for the termination of the ALLIIS algorithm suffices, since sets added to TBP is a subset of those added under ALLIIS. □

**Theorem 3.2.4.** If TBP is initialized as the set  $\{\text{Inv}(V)\}$ , then ALLIIS\_list produced from ALLIIS with LookUp will contain all the isolated invariant sets in  $V$  by the termination of ALLIIS with LookUp.

*Proof.* Let  $V'$  be some set pulled from TBP at Step 1. If  $V' = \text{Inv}(o(V'))$ , then  $V'$  is placed in ALLIIS\_list in Step 2.

Claim:  $S$  is not lost.

Case 1: For some  $S_i \in \text{LookUpTable}$ ,  $S_i \subset V'$  and  $\text{IIS}(S_i) \not\subseteq V'$ . By Corollary 3.2.2,  $S \subseteq \text{Inv}(V' - \{v\})$  for some  $v \in S_i$ . So,  $\{\text{Inv}(V' - \{v\}) \mid v \in S_i\}$  is added to TBP.

Case 2: No such  $S_i$  exists. So,  $\{\text{Inv}(V' - \{v\}) \mid v \in V'\}$  is added to TBP. By Corollary 3.1.3,  $S \subseteq \text{Inv}(V' - \{v\})$  for some  $v \in V'$ .

In other words,  $S$  is not lost. Since, TBP terminates by Lemma 3.2.3, we have that ALLIIS\_list contains all isolated invariant sets in  $V$ .  $\square$

# Chapter 4

## Results and Discussion

### 4.1 Application of Algorithms on $\mathcal{F}_{selfloop}$

The algorithms from Computational Dynamics Software (CDS), [3], and the algorithms discussed in Chapter 3 are implemented on the examples we have constructed from Chapter 2, namely  $\mathcal{F}_{selfloop}$ ,  $\mathcal{F}_{complete}$ , and  $\mathcal{F}_{tent}$ . The results of this study demonstrates the effectiveness of each algorithm. As expected,  $\mathcal{F}_{selfloop}$  is a good example of the effectiveness of the ALLIIS algorithm over the power set approach. Consider, a grid of four elements as described in Chapter 2. The power set method would search through  $2^4$  sets. However, we can see that ALLIIS considers fewer sets. Table 4.1 shows how ALLIIS processes  $\mathcal{F}_{selfloop}$  with four elements, where parent is the set being processed.

Step of While-Loop	State of TBP	Current Parent	Children	ALLIIS_list
0	$\{\{1\}\}$	—	—	$\emptyset$
1	$\emptyset$	$\{1\}$	$\emptyset$	$[\{1\}]$

Table 4.1: ALLIIS Processes  $\mathcal{F}_{selfloop}$

As we can see, the algorithm considers the set  $\text{Inv}(\{0, 1, 2, 3\}) = \{1\}$ . This set happens to be an isolated invariant set, so it is placed in ALLIIS\_list. It has no children (the subsets formed in Step 3), so the algorithm terminates after one step. As discussed in Chapter 2, there is only one isolated invariant set in  $\mathcal{F}_{selfloop}$  at any given resolution. Thus, in this example, we only consider one set as opposed to  $2^{|V|}$  sets under the power set approach.

## 4.2 Application of Algorithms on $\mathcal{F}_{complete}$

Now, we can consider a very different extreme case,  $\mathcal{F}_{complete}$ . This example highlights the effectiveness of the LookUp Table. First, we implement ALLIIS. We begin at Step 1 by pulling from the TBP,

$$\text{Inv}(V) = \{0, 1, 2, 3\}.$$

Since this set is isolated, we place it in ALLIIS\_list, Step 2. In Step 3, we form the children of  $\{0, 1, 2, 3\}$ , which are

$$\{1, 2, 3\}, \{0, 2, 3\}, \{0, 1, 3\}, \{0, 1, 2\}.$$

Each of these sets are placed in TBP. Step 4 sends us back to Step 1. Now, we process  $\{1, 2, 3\}$ . However, this set is not isolated, since

$$\text{Inv}(\{1, 2, 3\}) = \{1, 2, 3\} \neq \text{Inv}(o(\{1, 2, 3\})) = \text{Inv}(\{0, 1, 2, 3\}) = \{0, 1, 2, 3\}.$$

So, it is not added to ALLIIS\_list in Step 2. At Step 3, we consider the children of  $\{1, 2, 3\}$ , which are

$$\{2, 3\}, \{1, 3\}, \{1, 2\}.$$

They are not placed in TBP at this moment, since each set is a subset of a set already in TBP. At this point, we notice that for any set  $S \subset \{0, 1, 2, 3\}$ , we have  $\text{Inv}(S) = S \neq \text{Inv}(o(S)) = o(S)$ . Thus, none of those sets will satisfy the isolation condition. The algorithm will consider  $2^4 - 1$  sets by the time the algorithm terminates, since every subset (except the empty set) is added to TBP.  $\mathcal{F}_{complete}$  is an extreme example, but it shows the weakness of ALLIIS when processing highly connected sets. To avoid this problem, we introduce a LookUp Table of pre-processed isolated invariant sets.

$S_i$	IIS( $S_i$ )
$\{0\}$	$\{0, 1, 2, 3\}$
$\{1\}$	$\{0, 1, 2, 3\}$
$\{2\}$	$\{0, 1, 2, 3\}$
$\{3\}$	$\{0, 1, 2, 3\}$

Table 4.2: LookUp Table for Complete Graph at depth 2

ALLIIS with LookUp begin at Step 1 with

$$\text{Inv}(V) = \{0, 1, 2, 3\},$$

which is placed in ALLIIS\_list at Step 2. At Step 3, we consider the children,

$$\{1, 2, 3\}, \{0, 1, 3\}, \{0, 1, 2\}.$$

Each of these sets are placed in TBP. Step 4 sends us back to Step 1. In Step 2,  $\{1, 2, 3\}$  is not added to ALLIIS\_list, since it is not isolated. In Step 3, we test  $\{1, 2, 3\}$  against the LookUp Table in Table 4.2 and find that  $\{1\} \subset \{1, 2, 3\}$ , but  $\text{IIS}(\{1\}) = \{0, 1, 2, 3\} \not\subset \{1, 2, 3\}$ . So, we apply a directed cut (Corollary 3.2.2/Step 3 of ALLIIS with LookUp) as necessary. The only child is  $\{2, 3\}$  and is added to TBP. Similarly, for the other sets in TBP, we will obtain  $\{1, 3\}$  and  $\{1, 2\}$ , respectively. Now, we consider  $\{2, 3\}$ . After applying directed cuts, the child is  $\{3\}$ . Eventually, the algorithm terminates after processing the last set,  $\{2\}$ . Overall, the algorithm terminated after 8 steps. We considered only 8 sets when using ALLIIS with LookUp, instead of  $2^4 - 1$  sets using ALLIIS without LookUp.

### 4.3 Application of Algorithms on $\mathcal{F}_{tent}$

Lastly, we will look at how ALLIIS and ALLIIS with LookUp behave on  $\mathcal{F}_{tent}$ . The tent map is a more interesting dynamical system, with more complicated dynamics. Figure 4.1 compares the performance of ALLIIS and ALLIIS with LookUp against the power set. We can see that as we increase our resolution, TBP increasingly becomes a smaller fraction of the power set. Even better, TBP in ALLIIS with LookUp is significantly smaller. Note, for book-keeping, we only add a child when it is not already a subset of some set in TBP.

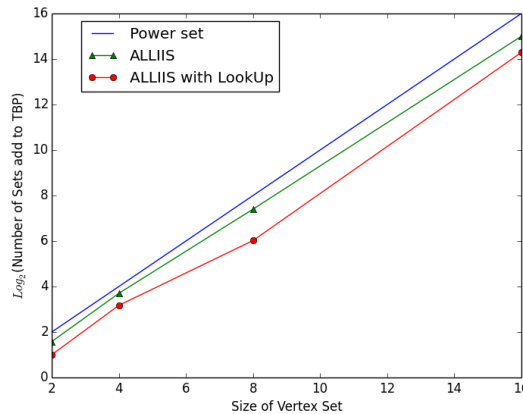


Figure 4.1:  $\text{Log}_2(\text{number of sets added to TBP})$  vs size of vertex set.  $\mathcal{F}_{tent}$  for  $|V| = 2, 4, 8, 16$ . The values at each grid resolution  $|V| = 2, 4, 8, 16$  are marked.

The tent map with parameter  $r = 2$  will only have three self-loops at resolutions of 8 elements or greater. So, the LookUp Table will only have at most three self-loops. Thus, one explanation of why the the difference between ALLIIS and ALLIIS with LookUp does not continue to increase could be that the LookUp Table is only seeded with self-loops. Thus, the number of directed cuts increases at a much slower rate than the increase in elements. If we consider the difference between ALLIIS and ALLIIS with LookUp (Figure 4.2), we can see this more clearly.

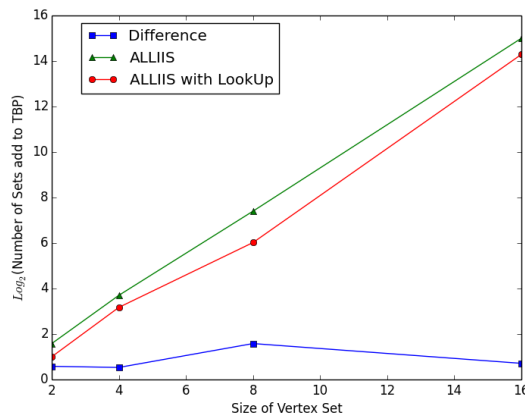


Figure 4.2:  $\text{Log}_2(\text{number of sets added to TBP})$  vs size of vertex set.  $\mathcal{F}_{tent}$  for  $|V| = 2, 4, 8, 16$ . The values at each grid resolution  $|V| = 2, 4, 8, 16$  are marked. The difference between ALLIIS and ALLIIS with LookUp does not increase with the increase in elements.

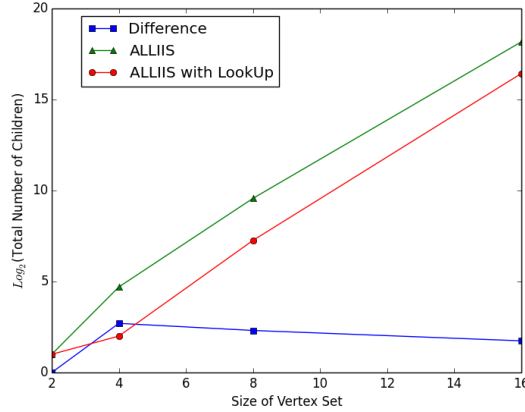


Figure 4.3:  $\text{Log}_2(\text{Total Number of Children})$  vs size of vertex set.  $\mathcal{F}_{tent}$  for  $|V| = 2, 4, 8, 16$ . The values at each grid resolution  $|V| = 2, 4, 8, 16$  are marked. The difference between ALLIIS and ALLIIS with LookUp does not increase with the increase in elements.

We can see this phenomenon, again, in Figure 4.3. ALLIIS (in green) considers more children than ALLIIS with LookUp (in red), and the advantage of ALLIIS with LookUp does not continue to increase. The difference between the two algorithms ceases to increase. So, the total number of sets that we process is still increasing exponentially.

Resolution $ V $	Runtime of ALLIIS (seconds)	Runtime of LookUp (seconds)
2	<1	<1
4	<1	<1
8	1	<1
16	667	189

Table 4.3: Comparison of runtimes between ALLIIS and ALLIIS with LookUP for  $\mathcal{F}_{tent}$  for  $|V| = 2, 4, 8, 16$ .

The runtime also increases greatly as seen in Table 4.3. Attempts at running the algorithms at resolution  $|V| = 32$  failed to halt after 12 hours.



Resolution $ V $	Isolated Invariant Sets
2	$\{0, 1\}$
4	$\{0, 1, 2, 3\}$
8	$\{0, 1\}, \{0, 1, 2, 3, 4, 5, 6, 7\}$
16	$\{0, 1\}, \{9, 10, 11\}, \{0, 1, 9, 10, 11\},$ $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

Table 4.4: Resulting isolated invariant sets for  $\mathcal{F}_{tent}$  for  $|V| = 2, 4, 8, 16$ .

Finally, both algorithms return the same output. Table 4.4 shows the isolated invariant sets found by these algorithms at each resolution. These isolated invariant sets can also be verified by computing by hand. As we increase our resolution, we should expect more isolated invariant sets for the tent map at parameter  $r = 2$ . With the isolated invariant sets computed, we may use Conley Index theory to prove things about the dynamics of the tent map. For example at a resolution of 16 elements, we can already determine, via arguments from the theory, that there exists a fixed point in the interval corresponding to  $\{0\}$  and the interval corresponding to  $\{10\}$ . We can verify that this is true, analytically (see [2]). Improving the LookUp Table and directed cuts, it should be feasible to compute the isolated invariant sets for depths that are too high to compute by hand and determine more complicated dynamics.

## 4.4 Discussion

With the limited success of ALLIIS and ALLIIS with LookUp algorithms, we immediately ask if there is room for improvement. If we were to seed the LookUp Table with more recurrent sets, such as 2-cycles, 3-cycles, etc., we should expect the difference between ALLIIS and ALLIIS with LookUp to increase, since we are applying more direct cuts. I speculate that there is a relation between the resolution,  $|V|$ , and the recurrent sets,  $|V|$ -cycles that should seed the LookUp Table to optimize this algorithm for the tent map. Work in determining such a relationship may also be beneficial to the application of ALLIIS with LookUp to other maps. This would allow us to further investigate how the performance of the algorithms scale with the resolution.

# Bibliography

- [1] Alligood, Kathleen, Tim Sauer, and James Yorke. *Chaos: An Introduction to Dynamical Systems*. New York: Springer, 1996. Print.
- [2] Kaczynski, Tomasz, and Konstantin Michael Mischaikow. *Computational homology*. New York: Springer, 2004. Print.
- [3] Kalies, W.D. *Computational Dynamics Software 2.1*. Florida Atlantic University, 2015.
- [4] Sarah Day, Rafael Frongillo, and Rodrigo Trevino. *Algorithms for Rigorous Entropy Bounds and Symbolic Dynamics*.