CHAOTIC TRAVERSAL (CHAT): VERY LARGE GRAPHS TRAVERSAL USING CHAOTIC DYNAMICS

Boonyarit Changaival*
FSTC/CSC-ILIAS, University of Luxembourg,
6, Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
boonyarit.changaival@uni.lu

Martin Rosalie
SnT, University of Luxembourg
6, Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
martin.rosalie@uni.lu

Grégoire Danoy
FSTC/CSC-ILIAS, University of Luxembourg,
6, Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
gregoire.danoy@uni.lu

Kittichai Lavangnananda
School of Information Technology, King Mongkut's University of Technology Thonburi,
126 Pracha Uthit Rd., Bang Mod, Thung Khru, Bangkok 10140, Thailand
kitt@sit.kmutt.ac.th

Pascal Bouvry
FSTC/CSC-ILIAS/SnT, University of Luxembourg,
6, Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
pascal.bouvry@uni.lu

Graph Traversal algorithms can find their applications in various fields such as routing problems, natural language processing or even database querying. The exploration can be considered as a first stepping stone into knowledge extraction from the graph which is now a popular topic. Classical solutions such as Breadth First Search (BFS) and Depth First Search (DFS) require huge amounts of memory for exploring very large graph. In this research, we present a novel memoryless graph traversal algorithm, Chaotic Traversal (CHAT) which integrates chaotic dynamics to traverse large unknown graphs via the Lozi map and the Rössler system. To compare various dynamics effects on our algorithm, we present an original way to perform the exploration of a parameter space using a bifurcation diagram with respect to the topological structure of attractors. The resulting algorithm is an efficient and non-resource demanding algorithm, and is therefore very suitable for partial traversal of very large and/or unknown environment graphs. CHAT performance using Lozi map is proven superior than the, commonly known, Random Walk, in terms of number of nodes visited (coverage percentage) and computation time where the environment is unknown and memory usage is restricted.

Keywords: Bifurcation Diagram, Chaotic Dynamics, Graph Traversal, Random Walk

*Corresponding author

1. Introduction

Graphs are used nowadays for knowledge representation in biology, social network and World Wide Web [Khan & Elnikety, 2014; Lovász, 2012]. Coupling with the "Big Data" era, the size of these graphs can be extremely large. A very prominent example of large graph is the Internet which was estimated to be over 30 billion nodes. Another interesting example from biology is the human brain which consisted of 10^{11} nodes [Lovász, 2012]. These numbers were just estimations. In reality, in such huge graphs, their available information is never complete, or even worse, completely unknown [Lovász, 2012]. A lot of efforts were also put into large graph analysis as the application can cause a great impact on many communities. Taking a benchmark suite called "Graph500" as an example, it contains a graph with around 67 million nodes [Murphy *et al.*, 2006] and this benchmark has been widely used as another index to test "High Performance Computing" (HPC) clusters performance. However, unlike previously mentioned large graphs, this graph was synthesized and all the information was known after the generating process according to Lovász [2012]. As large real world graphs are not completely known, graph exploration or graph traversal algorithms are needed to learn more about the graphs in order to extract meaningful information. Therefore, graph traversal is the first step for knowledge extraction from the real world graphs. In this respect, the focus of this paper aims at a large graph discovery. We focus on large graphs where in nature, global information and properties are unknown prior to the discovery.

A Graph G is a set of vertices and edges defined as follows $G = (V, E)$. Edges are unweighted and undirected. The Graph Traversal problem in this work is formulated similarly to a problem introduced by Kalyanasundaram. & Pruhs [1994] which created the shortest tour based on local decisions. We call an *agent* the entity visiting the various nodes of the graph. Under this formulation, the agent does not have any information about a topology of the graph nor its size, it can only learn about the neighbors of the vertex v when it visits the vertex. This is described as the "fixed graph scenario" [Kalyanasundaram. & Pruhs, 1994]. The agent starts from an arbitrary (random) node and traverses to the next node crossing an existing edge. Classical termination criteria for the graph traversal problem are when all the nodes or links have been fully discovered or when a tour has been constructed which relies on the fact that the traversed graph is known and memory is unlimited [Panaite & Pelc, 1999; Kalyanasundaram. & Pruhs, 1994; Albers & Henzinger, 2000]. However, in our scenario this criterion is not applicable considering that the graph size is unknown. Hence, in order to compare the exploration performance of the algorithms we use a Coverage Percentage which is the ratio of discovered nodes after a predefined number of iterations to the number of nodes in the tested graphs. Here, we formulate our research questions, i.e. what is the best agent strategy for node selection given that the agent is memoryless and that the objective is to maximize the discovery of unvisited nodes at each step.

Generally, algorithms using random numbers are used to solve the graph traversal problem [Coppersmith *et al.*, 1993; Fleischer & Trippen, 2003]. This gives rise to a question whether there are any other strategies to solve this problem. As a result, our research on chaotic dynamics is conducted. In the optimization field, recent works include chaotic dynamics to replace the random part of algorithms to enhance their performance [Bucolo *et al.*, 2002; Li *et al.*, 2006]. For instance, considering the "Particle Swarm Optimization" (PSO) algorithm, chaotic maps are used for the diversification: logistic maps [Liu *et al.*, 2005; Xiang *et al.*, 2007] or more complicated maps [Araujo & Coelho, 2008; Gandomi *et al.*, 2013b; Pluhacek *et al.*, 2015]. In this work, we select two chaotic dynamics, namely, Lozi map and Rössler system (more detail in Section 3). The complicated dynamics obtained from both systems are explored and the study on their effects on the traversal problem is also conducted. Moreover, the in-depth study of the Rössler system is also made possible as there are already tools to compare these non equivalent dynamics using a bifurcation diagram [Rosalie, 2016].

In this paper, we present a novel memoryless agent-based graph traversal algorithm which integrates the chaotic dynamical system. We test Lozi map and Rössler system with our algorithm to show the compatibility and prospect of chaotic dynamics in a graph traversal problem. We also present an original method to explore the performance of an algorithm using a bifurcation diagram (of Rössler system) to variate a parameter with respect to the topological properties of the attractors. The remainder of this article is organized as followed. Section 2 defines the Graph Traversal problem and the graph topologies;

it also presents the related state of the art. In Section 3, the notion of Chaotic Dynamics is explained with details about the bifurcation diagram we use. Then a novel graph traversal algorithm is described in Section 4. Section 5 is dedicated to the study of the influence of the parameter α in Rössler system and experimental result. In Section 6, the discussion on the parameters and experimental results of Lozi map are presented. Section 7 reports the Mean Time Coverage of CHAT algorithms in comparison to the Random Walk. We then summarize all of the results in section 8. Finally, we present the conclusion along with the future work in section 9.

2. Graph Topologies and Graph Traversal Algorithms

We first describe the topological structure of graphs used in literature to highlight the potential application of our algorithm in various fields and then we provide a state of the art of the graph traversal algorithm.

2.1. *Graph topologies*

In this research, we test our algorithm on five topologies; ring, small world, random, grid and power-law because these topologies are found widely in the applications in the real world. We give some brief example usages of each topology to establish the understanding of wide applications for these graphs. Starting from the Ring topology, ring topology can be found mainly in Network field as it is relatively simple and reduce the packet collision in the network. This kind of network also allows the growth in the network without impacting the performance. An obvious example to show that ring network can be large is the "Metropolitan Area Network" (MAN). This network has to accommodate to the increase usage of traffic data in the metropolitan area. There were some projects that proposed the architecture of the MAN in a ring topology which were HORNET [Shrikhande *et al.*, 2000], RINGO [Gaudino *et al.*, 2001] and KOMNET [Richter *et al.*, 2001].

The second topology is a small world Topology. Six degree of separation is the main concept of the small world topology. It means that each node or entity in the graph can be reached within six hops (on average) from any nodes[Horvitz & Leskovec, 2007]. This topology is prominence in the social network graphs where the small world properties can be found [Watts & Strogatz, 1998]. It was also shown in [Adamic, 1999] that the World Wide Web (WWW) has the small world properties. In addition, the small world effect also inhibits in biological networks such as "Protein-Protein Interaction" (PPI) [Yook *et al.*, 2004] and "Metabolic Network" [Wagner & Fell, 2001] and in infrastructure networks such as the railway network mentioned in [Seaton & Hackett, 2004]. An interesting use case is the use of the loss of small world effect in the brain graph to analyze the Alzheimer disease in [Sanz-Arigita *et al.*, 2010]. This demonstrates that the small world effect actually exists in the real world problems.

The next topology is the topology that appeared in many works: random topology. One of the most popular random graph is Erdős-Rényi random graph proposed in [Erdős & Rényi, 1960]. The random graph might lack the transitivity and has unrealistic Poisson degree distribution, but it is a staple for graph problems since it is one of the graphs that has been used to test algorithms [Lovász, 2012; Servetto & Barrenechea, 2002; Haenggi *et al.*, 2009]. Not to mention that random graphs with some degree distributions, can be used to predict and model the real world graph as well [Newman *et al.*, 2001]. This property makes random topology a good choice for a preliminary testing of an algorithm. That is not all to the random graph. In [Wong & You, 1985], the authors use the merging of random graphs to generate patterns and use them in classification. The random graph was also introduced as a model to analyze the programming code and refined it to have a low error-rate [Alon *et al.*, 1992].

Another topology that is heavily used in robotic field is a grid topology. Grid topology is used as a problem model in covering problem. The plain is usually modeled in grid with or without obstructions [Moravec & Elfes, 1985; Koenig *et al.*, 2001; Yanovski *et al.*, 2001; Rosalie *et al.*, 2016]. Even though, in the mentioned work, they used "Grid Map", the map can be translated into a graph without any loss of information. It is for this reason that we include this topology into our experiment to test the versatility of our algorithm.

The last topology is power-law topology. Power-Law graphs have the degree distribution that conforms to the power-law distribution [Mitzenmacher, 2004; Holme & Kim, 2002]. The property of power-law

topology can be found in several real world network such as the Internet [Albert *et al.*, 1999; Faloutsos *et al.*, 1999] and biology graphs. In fact, past research pointed out that power-law distribution exists in many biological graphs such as Yeast-Protein Interaction networks, Metabolic network [Barabasi & Oltvai, 2004], "Protein-Protein Interaction" (PPI) networks [Yook *et al.*, 2004] and Cellular networks [Albert, 2005]. From these findings, we would like to point out that it is common to find graphs that contain several properties (from various topologies) such as a PPI network where a small world effect can be found along with the power-law distribution. Moreover, these networks are growing each day due to the new discoveries, even existing ones are already large, e.g. brain network that contains a few hundred billions nodes [Lovász, 2012].

Given the applications of each topology and their associations with the real world use cases, there is a possibility that these graphs can be large and contains a lot of useful information. These topologies represent real problems in various fields; networking, social network analysis, biology, and robotic. Furthermore, It is not always the case that these graphs are known beforehand since they are growing rapidly due to the age of discovery and become very hard to track. hence, these are reasons why we choose to incorporate these topologies in our experiment to demonstrate the potential of application in the real world.

2.2. *Graph Traversal Algorithms*

The purpose of the graph traversal problem is to discover vertices (or nodes) in the graph and at the end, learn about the information that a graph represents. Graphs can be simply classified into two types based on their available information, i.e. known and unknown graphs. In the case of known graph, the global information of the graph, such as the size or the topology, is available beforehand. An algorithm can then exploit this knowledge to efficiently execute its traversal on the graph. On the other hand in an unknown graph, no information is available which makes it more difficult to traverse it. In this work, we are interested in the latter problem with the aim of using no memory or as minimal memory as possible.

The unknown graph traversal mostly finds its roots in the robotic field as the robot needs to explore an unknown terrain and is known as the covering problem [Panaite & Pelc, 1999; Albers & Henzinger, 2000; Choset, 2001; Dudek *et al.*, 1991]. It was started with a Random Walk algorithm which let a robot choose the next destination randomly through a uniform random function [Coppersmith *et al.*, 1993; Fleischer & Trippen, 2003]. Then, the heuristic algorithm called the "Nearest Neighbor Approach" [Koenig & Smirnov, 1996] was introduced. It is a greedy algorithm that creates the shortest path from the current node to the destination based on the retrieved local information (cost or weight) of the current node. After that, more heuristic algorithms: Depth First Search (DFS) [Kwek, 1997] and Breadth First Search (BFS) [Albers & Henzinger, 2000] were introduced. Breadth First Search and Depth First Search performed extremely well on known graphs, however, they fell short in the unknown graph traversal and both need an additional feature called relocation. This feature relocated the starting point to the nearest neighbor of the previous starting point to prevent the algorithm from being stuck in a loop. In [Fleischer & Trippen, 2003], the authors showed the performance comparison between those algorithms on connected directed random graphs. They also pointed out that randomized algorithms actually did not compromised any performance comparing to DFS and BFS algorithms. Yet, these works were tested on small graphs only. The memory issue will show itself when algorithms like BFS and DFS are used to traverse the graph as the required memory for both BFS and DFS algorithms is $O(b^{d+1})$ where b is the average out-degree and d is the distance from the starting node. This distance d can be viewed in the same fashion as hops from the starting node or levels in a tree graph from the root [Fraigniaud *et al.*, 2005].

Even though memory is a big issue in a big graph traversal, it is not the only one; traversing time is also an issue. Several studies contributed to parallelize those algorithms. For example, in [Hong *et al.*, 2011], the author parallelized Breadth First Search (BFS) on Central Processing Unit (CPU) and Graphic Processing Unit (GPU) and also proposed a hybrid method which alternates between sequential, CPU and GPU implementations depending on the situation. However, it can be applied on known graph only which is a primary constraint of the BFS algorithm since the algorithm need to know when all nodes are discovered to stop the search which is not the case in unknown graph scenarios. Looking into an unknown graph traversal problem, agent-based systems are more widespread since they require little resource and relatively

easy to parallelize [Wagner *et al.*, 1999; Koenig *et al.*, 2001; Yanovski *et al.*, 2001]. For example, Ilcinkas [2008] proposed a "Port Selection Technique" which requires each node to have a set of memory. Then, this memory is used to remember which port has already been selected by agents and not to be repeated. The algorithm can also accommodate multiple agents. Apart from this technique, ant-based algorithms are also popular. Wagner *et al.* [1999] proposed a covering method based on ants and pheromones. The pheromones were used to enhance the exploration. They proposed and compared two approaches: the first one leaves pheromones on edges and the other one leaves pheromones on vertices. The result showed that in the latter, the covering time was significantly lower. The authors also argued that this method is efficient even for dynamic graphs where edges or nodes can be altered unlike the DFS. Koenig *et al.* [2001] proposed a repeated coverage method using ant robot. The authors experimented on the greedy method in which the agent always take the node with the least visiting number as the next destination. Another method is called Learning Real-Time A* (LRTA*) where the agent calculates the attractiveness of the node based on its unvisited or least visited neighbors. From the experiment, the author concluded that in most situations, LRTA* is more efficient than a greedy algorithm. Yanovski *et al.* [2001] proposed a simple multi-agent system to keep patrolling (continuous exploration) and aimed to achieve a uniform visiting frequency for all edges. As there are many classes of graph, Higashikawa *et al.* [2014] also proposed the agent-based graph traversal algorithm called "BEER" on cyclic and tree graphs. The authors also pointed out that a greedy algorithm can still work very well on tree without any need of sophisticated methods. Still, all aforementioned approaches need memory to store the already selected paths or the pheromones on all vertices.

Some examples of graph traversal algorithm applications can be found in [Dentler *et al.*, 2009; Tiddi *et al.*, 2014]. Dentler *et al.* [2009] proposed an ant colony based algorithm to create pattern-based inference rules on the Resource Description Framework (RDF) Graph for Semantic Web Reasoning whereas the ant-based system performed better than a random strategy. To obtain the inference rules, ants needed to traverse the graph first and build rules simultaneously making traversal algorithm a crucial part of the process. There is another work related to knowledge discovery: Tiddi *et al.* [2014] proposed "Linked Data Traversal" which uncovered the unknown graph on real time. In addition, the traversal problem is also studied in the dynamic graph domain where the graph is changed over time. In [Piyatumrong *et al.*, 2009a,b], the authors used the token traversal method to find the network backbone in a dynamic ad-hoc network. The algorithm only utilized a single hop knowledge to determine the next node to traverse which made it completely based on local information. This idea is very suitable for a large unknown graph. On that account, the application of various graph traversal algorithms can potentially be extended to more of the real world use cases.

Due to the fact that chaotic dynamics had been reported to improve the performance of optimization algorithms [Bucolo *et al.*, 2002; Gandomi *et al.*, 2013a,b] and covering algorithm [Rosalie *et al.*, 2016] over the usage of a typical uniform random function, we intend to test it against a Random Walk algorithm which is memoryless and employs a uniform random function as well to study the result and the chaotic dynamic contribution on the graph traversal problem enhancement. Moreover, with the deterministic property of a chaotic dynamic, it also promotes reproducibility of the work. From the best of authors knowledge, there is not yet a work that integrates chaotic behavior into the graph traversal algorithm and requires no memory or constant memory to process. Therefore, this research broaden the horizon of chaos theory. In the next section, chaotic behavior and dynamics are elaborated in order to explain the algorithm of such a unique character.

3. Chaotic Dynamics

As defined by the Encyclopedia Britannica: "Chaos theory, in mechanics and mathematics, is the study of apparently random or unpredictable behavior in systems governed by deterministic laws.". Chaos theory is not only limited to the study of the system alone, but also had long been applied in many problems such as optimization problems [Bucolo *et al.*, 2002] where the chaotic maps were tested against Random process. An improvement in performance was also reported. In effect, chaotic systems were introduced in several works related to optimization processes such as [Araujo & Coelho, 2008; Liu & Chen, 2010; Gandomi *et al.*,

2013a] where the uniform random function was replaced by the chaotic system and performance improved. However, the optimization is not the only field that has seen a use of chaotic dynamics, it can also be used to generate random numbers as appeared in [Lozi, 2012] where the Lozi map is used for "Chaotic Pseudo Random Number Generator" (CPRNG). Hence, there is another kind of work where, instead of utilizing chaotic map directly, CPRNG is used in place of the typical uniform random number generator, for example, Pluhacek *et al.* [2015] fine-tuned Lozi map and CPRNG, and improved the performance of "Particle Swarm Optimization" (PSO) algorithm over the one that employed the uniform random function. Another recent work which employed chaotic dynamics was the coverage problem by [Rosalie *et al.*, 2016]. The authors integrated the chaotic behavior to enhance the drones' searching capability. The chaotic behavior has demonstrated its ability to enhance the algorithmic performance over purely random behavior in optimization and covering problems [Araujo & Coelho, 2008; Bucolo *et al.*, 2002; Gandomi *et al.*, 2013a; Liu & Chen, 2010; Rosalie *et al.*, 2016]. With similarity between a covering problem and a graph traversal problem along with studies that confirmed an improved performance of chaotic systems over a random behavior, a chaotic system is used as a based for our proposed graph traversal algorithm. We use two chaotic dynamics in our algorithm: the first one is the Lozi Map and the second one is the Rössler System. Lozi map is a simple map which contains only piecewise linear equations in two-dimensional space while, the Rössler System is defined by three differential nonlinear equations in three-dimensional space. The two dynamics are introduced in this section.

3.1. *Solution of a Dynamical System and its Properties*

The deterministic chaotic solution of a system has three important properties, these are:

- globally time invariant;
- highly sensitive to the initial conditions;
- aperiodic.

The first property implies that, given the same parameters, the attractor at $t = 0 \rightarrow t = 10$ is the same as the attractor at $t = n \rightarrow t = n + 10$. The second property implies that even considering a difference in the scale of 10^{-10} for the initial conditions, the system will divert from the original track. The last property points out that the chaotic system always gives out non-periodic solution. However, it is important to note that the solution can be chaotic or periodic: it depends on the system's parameters.

In this paper, we experiment with two chaotic dynamics; Lozi map [Lozi, 1978] and Rössler system [Rössler, 1976]. Lozi map is a discrete chaotic map made of two dimensions which depends only on two parameters. While, on the other hand, Rössler system is a continuous system of three differential equations (three dimensions) that depends on three parameters.

3.1.1. *Lozi Map*

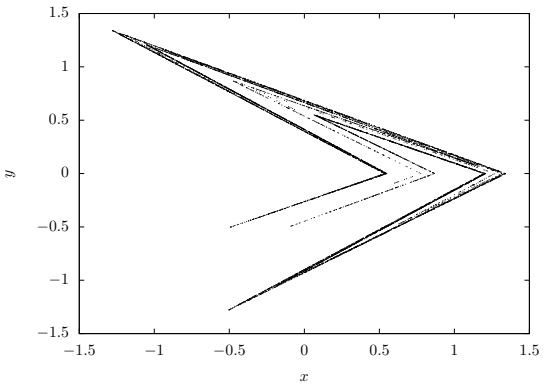


Fig. 1. Lozi attractor with the parameters $a = 1.7$ and $b = 0.5$ solution to Eq. (1).

We choose to employ the Lozi map since it is also relatively simple by its definition, yet offer a certain degree of alteration to the behavior of the dynamics. The Lozi attractor is a strange attractor similar to the Hénon attractor. It is defined by two equations in Eq. (1) [Lozi, 1978].

$$\begin{cases} x_{n+1} = 1 - a|x_n| + y_n \\ y_{n+1} = bx_n \ . \end{cases} \tag{1}$$

For the Lozi attractor, x and y are variables. At time $t = 0$, these variables are also called initial conditions. The attractor is modified through the parameter a and b. The Lozi attractor with $a = 1.7$ and $b = 0.5$ is shown in Fig. 1. For further reading, the reader are referred to [Botella-Soler *et al.*, 2011] for in-depth analysis of the behavior of Lozi map.

3.1.2. *Rössler System*

First of all, it is important to note that in our work, Rössler system is defined by three "Ordinary Differential Equations" (ODE). This aspect makes the Rössler system in our work a continuous system. There are also tools to solve ODE from several fields that deal with nonlinear systems and we choose to use "Runge-Kutta method (fourth order)" (RK4). We are also aware that this step increases the computation time. However, combining it with the Poincaré section (discretization method), we can analyze chaotic dynamics which is needed in our work since we are aiming to study the link between chaotic dynamics and graph traversal problem. We then present three major steps for calculating the solution from a chaotic system which are explained in detail later in this section. These steps focus only on the outcome of the system, further details can be found in [Rosalie, 2016] for the Rössler system.

(1) Choose the parameters and initial conditions of the system that give an attractor as the solution for the system (Fig. 2).
(2) Discretize the system using Poincaré section (Eq. (4)).
(3) Compute the First Return Map (the dynamical signature of the system) (Fig. 3).

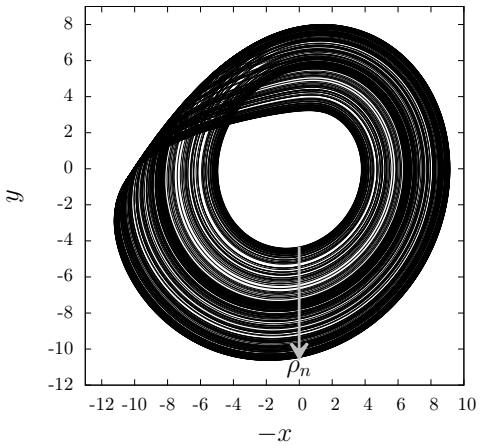


Fig. 2. Attractor solution to the Rössler system (Eq. (2)) for $\alpha = 0$ (Eq. (3)) with Poincaré section. The arrow indicates the orientation of the normalized value ρ_n.

The Rössler system [Rössler, 1976] is a three differential equations system:

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \ . \end{cases} \tag{2}$$

This system provides the solution which can be an attractor or a periodic solution. The first derivative $\dot{x}$, $\dot{y}$ and $\dot{z}$ is the definition of the problem where the x, y and z are the variables of the system where all variables at $t = 0$ are called initial conditions. A series of variables x, y and z from $t = 0 \rightarrow \infty$ is the

solution of the system. Lastly, a, b and c are parameters to alter the behavior of the system. However, [Sprott & Li, 2017] already came up with a method to generalize these a, b and c parameters modification with α as seen in:

$$\begin{cases} a = 0.2 + 0.09\alpha \\ b = 0.2 - 0.06\alpha \\ c = 5.7 - 1.18\alpha \ . \end{cases} \tag{3}$$

The bifurcation diagram (Fig. 4) illustrates the effect of how the parameter α can alter the system. In the next section, we describe it in detail. On the x-axis is the value of α. On the y-axis is the value of the solution in the Poincaré section

$$P \equiv \{(y_n, -z_n) \in \mathbb{R}^2 | x_n = x_-, -\dot{x} < 0\} \tag{4}$$

where x_- is the x value of the singular point of the Rössler system in the center of the attractor (see [Rosalie, 2016] for details). Such a parametrization gives a bifurcation diagram following the topological properties of the attractor (Fig. 2). From y_n in Eq. (4), we can then normalize it to obtain ρ_n and plot it to visualize the first return map (Fig. 3). The detailed method to obtain first return map and its application is explained in Section 3.2.

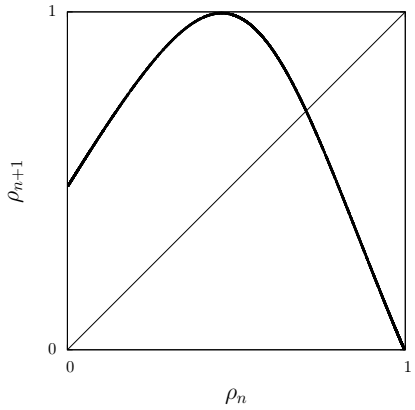


Fig. 3. First Return Map to the Poincaré section for $\alpha = -0.25$.

3.2. *Bifurcation Diagram for Optimization*

The bifurcation for the Rössler system and the Lozi map are presented in this section. In this work, the bifurcation diagram is used to represent states of a solution for each parameter whether it is periodic or chaotic. For the Rössler system, the parameters are generalized with α as shown in Eq. (3). We present in this section a Rössler system bifurcation diagram where $\alpha = [-0.5, 1.2]$. On the other hand, Lozi map has two parameters, a and b. We select two sets of parameters, $a = 1.5, b = [-0.5, 0.47]$ and $a = [1.1, 1.8], b = 0.1$. Both were mentioned in [Botella-Soler *et al.*, 2011],

3.2.1. *Rössler System Bifurcation Diagram*

A bifurcation diagram of the Rössler system is obtained through Poincaré Section. From the bifurcation diagram, we can roughly understand the behavior of the system at each α which means that we are able to provide first return maps. Then it enables us to proceed with the optimization of an algorithm afterward depending on first return maps. The bifurcation diagram of the Rössler system with $\alpha = [-0.8, 1.2]$ is shown in Fig. 4. There are dense periods and sparse periods. The dense periods are the chaotic attractors which are the focus of this work and the sparse periods are the periodic solutions where the solution alternates between few values. For example, at $\alpha = -1.5$, there is only one point and is called a period-one solution. Another example is $\alpha = -1$, there are only two points and is called a period-two solution. These two α

(and the likes) do not show the chaotic dynamic behavior. Thus, it can be clearly seen in Fig. 4 that by varying the value of α, chaotic or periodic solutions can be provided.

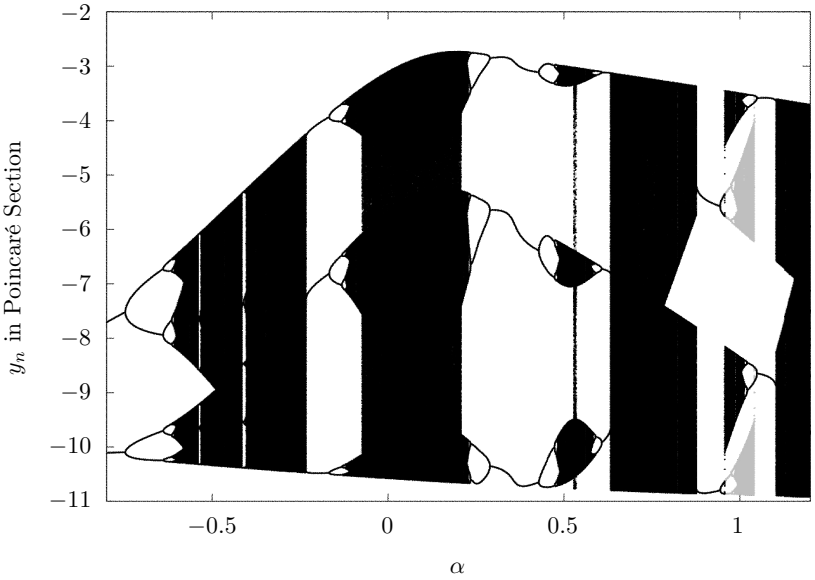


Fig. 4. The bifurcation diagram where α is varied in Eq. (3) for the Rössler system Eq. (2).

Regarding the chaotic solutions, it is also worth mentioning that there is a special case among chaotic solutions as well, called "Banded Chaos". It is an attractor or a solution that is composed of stripes or bands. In Fig. 4, around $\alpha = 1.1$, there is a gap in the middle which is different from the area where $\alpha = 0$ and that is an example of banded chaos area. As for the characteristic of the banded chaos, it can be regarded as a more complex than a normal chaotic attractor. Readers who would like to pursue further detail are referred to [Rosalie, 2016]. Regardless of chaotic or periodic solutions, once the system is solved we obtained a continuous solution.

For discretization, we use a Poincaré section as can be seen in Eq. (4). The interpolation between two sequential discretized points is calculated. This process is illustrated in Fig. 2. The line is drawn on the plane and cut though the solution and then the interpolation can be computed.

After the discretization, the computation of the first return map follows. The discretized values from Poincaré section are normalized to obtain $\rho_n \in [0, 1]$ from y_n:

$$\rho_n = (y_n - UB)/(LB - UB) \; . \tag{5}$$

The normalization is based on the bifurcation diagram and equations are used to estimate the upper bound and lower bound. However, bifurcation diagram is not in linear shape, thus we divide it into several parts to ensure that we retrieve values of ρ_n between 0 and 1. LB and UB of Eq. (5) can be found in Eq. (6) and Eq. (7) respectively. The partitions on the bifurcation diagram are also shown in Fig. 5.

$$LB = 0.110626\alpha^2 + 0.4141\alpha - 10.585 \tag{6}$$

$$UB = \begin{cases} 6.04162\alpha - 2.8 & \text{if } \alpha = [-0.5, -0.22] \\ -7.16446\alpha^2 + 3.31662\alpha - 3.05252 & \text{if } \alpha = (-0.22, 0.192] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (0.192, 0.790) \\ -4.31521\alpha - 4.01651 & \text{if } \alpha = [0.790, 0.876] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (0.876, 1.102) \\ 23.673\alpha - 34.2435 & \text{if } \alpha = [1.102, 1.155] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (1.155, 1.2] \end{cases} \tag{7}$$

In [Rosalie, 2016], it had been shown that the Rössler dynamics can be considered similar as a whole using templates and subtemplates. Thus, to generate non-equivalent dynamics, we have to create a partition

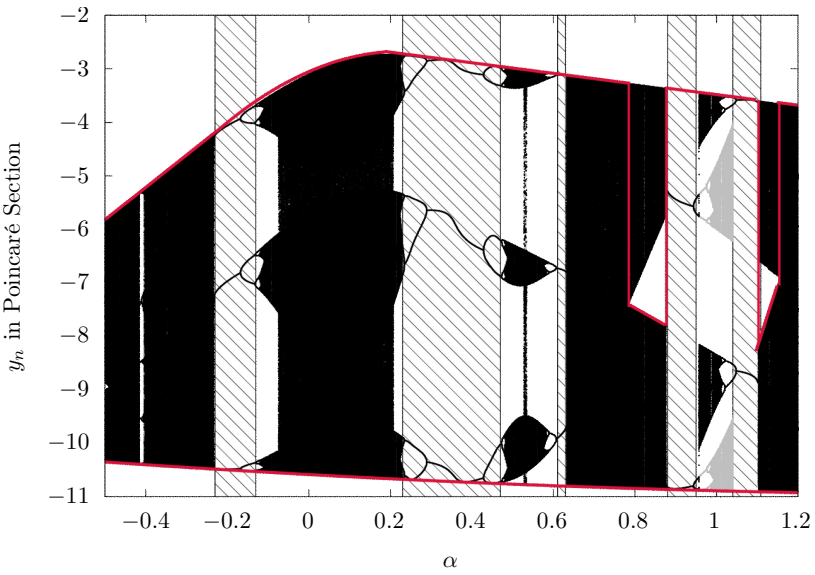


Fig. 5. Partitions in the bifurcation diagram according to Eq. (6) and Eq. (7)

out of a bifurcation diagram. Then, it is also very important to note that the fourth and sixth equation in Eq. (7) only considers one band from a whole banded chaos as appeared in [Rosalie, 2016] in first return maps section. Otherwise, if the whole bands are taken, it will result in discontinuous first return maps and shows a behavior that is similar to periodic solutions which is not our focus. However, this specific partition is not required for $\alpha = 1 \pm 0.05$ because the two co-existing attractors have the topological structure as it is for the beginning of the bifurcation diagram. The templates of the two co-existing attractors are symmetric by inversion for these values of α. When α increases, the templates are the same for the co-existing attractor when α decreases. In addition, these templates have the same topological structure of those observed at the beginning ($\alpha < -0.25$) of the bifurcation diagram: templates with two branches with one increasing and one decreasing [Rosalie, 2016]. Finally, there is a redundancy in the templates if we consider them all and the partition performed using UB in Eq. (7) permits to variate α to produce non equivalent dynamics.

The first return map which is a dynamical signature of the system can then be realized by plotting the current discretized solution against the next. The first return map can have many shapes depending on the provided α. Fig. 3 is just one of many possibilities available from the Rössler system. After an extensive study of the system, we found that the system itself, regardless of the first return map, incorporates prominent patterns. So far, we have found three prominent patterns that are related to the periodic orbits of the system that are visited more often.

(1) **Growing Pattern**: The value of the solution starts from a very low value and gradually increasing to the high value. Once the high value is reached, the solution value drops to low and the process start again as can be seen in Fig. 6.
(2) **Oscillation Pattern**: The value of the solution alternates between high and low values. The example is shown in Fig. 7. This patterns corresponds the period-2 orbit for Fig. 7.
(3) **Plateau Pattern**: There is little change in term of value of solution during this pattern period. The Fig. 8 might not be totally linear, but the change in value is still small. This value corresponds to the period-1 orbit of the system for Fig. 8.

It is important to note that each α yields a different first return map. This corresponds to the frequency of the patterns and as a consequence, a different behavior. In addition, these patterns correspond to the unstable periodic orbits of the attractor, but since the value of α is varied, we cannot provide more links between the orbits and the patterns (see Fig. 13 of [Rosalie, 2016] for more details about the dynamical partition of this bifurcation diagram).

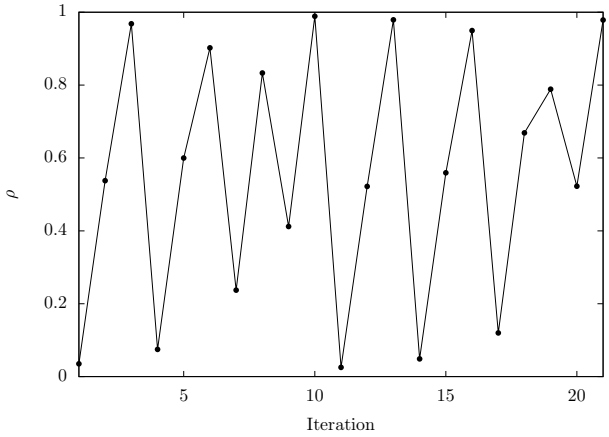


Fig. 6. A plot of ρ_n against iteration for growing pattern.

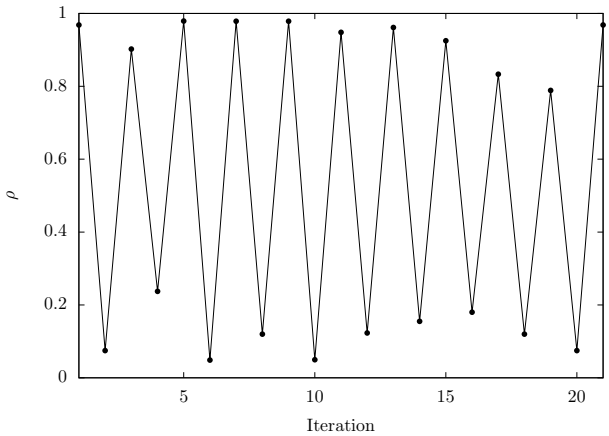


Fig. 7. A plot of ρ_n against iteration for oscillation pattern.

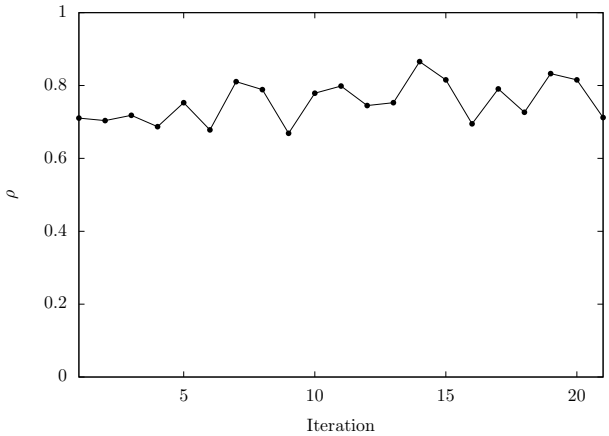


Fig. 8. A plot of ρ_n against iteration for plateau pattern.

3.2.2. *Lozi Map Bifurcation Diagram*

The bifurcation diagrams of the Lozi map used in this work are based on the value of x from the map. Two bifurcation diagrams of the selected parameters ($a = 1.5, b = [-0.5, 4.7]$ and $a = [1.1, 1.8], b = 0.1$) are shown in Fig. 9 and Fig. 10. As far as the authors know, there is no possible comparison between the chaotic mechanisms as it has been done for the Rössler system using topological analysis. Some works in

that direction are still in progress using suspension of the map [Starrett, 2012; Mangiarotti & Letellier, 2015]. Thus we only applied the same partitioning methodology in Section 3.2.1 to prevent the gap in the yielded x sequence. As a result from partitioning and Eq. (5) where x_n is used in place of y_n, the values of ρ_n from x_n are in $[0, 1]$.

The upper bound (UB) and lower bound (LB) in Fig. 9 for the Lozi map with $a = 1.5$ and $b = [-0.5, 0.47]$ are defined as follows;

$$LB = \begin{cases} -0.58728b^3 - 0.20915b^2 - 0.94836b - 0.50321 & \text{if } b = [-0.5, 0.4225] \\ 0.03116b - 0.831737 & \text{if } b = (0.4225, 0.47] \end{cases} \tag{8}$$

$$LB = \begin{cases} -4.82296b^2 - 1.47635b + 0.46621 & \text{if } b = [-0.5, -0.37) \\ 0.15227b^2 + 0.15227b^2 + 1.00494 & \text{if } b = [-0.37, 0.4225) \\ -2.55404b + 1.30011 & \text{if } b = [0.4225, 0.47] \end{cases} \tag{9}$$

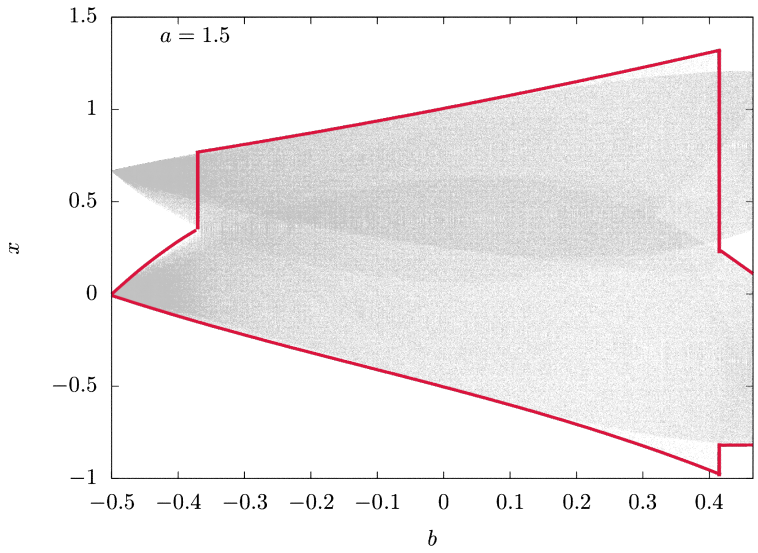


Fig. 9. Partitions in the bifurcation diagram according to Eq. (8) and Eq. (9) for the Lozi map with $a = 1.5$ and $b = [-0.5, 0.47]$.

Upper bound (UB) and lower bound (LB) in Fig. 10 for Lozi map with $a = [1.1, 1.8]$ and $b = 0.1$ are defined as follows;

$$LB = -1.00691a + 0.90294 \tag{10}$$

$$LB = \begin{cases} 6.31887a^2 - 14.4394a + 8.04677 & \text{if } a = [1.1, 1.23703) \\ 2.8679a^2 - 5.92552a + 3.05 & \text{if } a = [1.23703, 1.40334) \\ -0.05148a + 1.14403 & \text{if } b = [1.40334, 1.8] \end{cases} \tag{11}$$

From all presented bifurcation diagrams of the Lozi map, all solutions are chaotic. In contrast, it is not the case for the Rössler system. This emphasizes the importance of the bifurcation diagram as a preliminary filter for chaotic solutions. Moreover, with the partition method, more complex chaotic behaviors from the continuous chaotic system, e.g. Rössler system, can be explored. At the same time, it also benefits the discrete chaotic system, e.g. Lozi map, by eliminating the gap in the yielded sequence of ρ_n if needed.

4. Methodology for Graph Traversal Algorithms

We present a novel chaotic agent-based graph traversal algorithm that focuses on exploring unknown graphs called “Chaotic Traversal (CHAT)”. CHAT is actually similar to the Random Walk algorithm except that

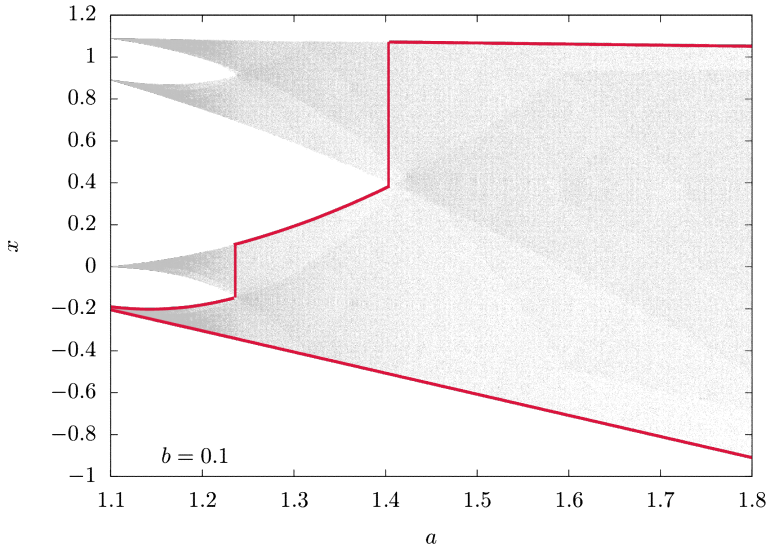


Fig. 10. Partitions in the bifurcation diagram according to Eq. (10) and Eq. (11) for Lozi map with $a = [1.1, 1.8]$ and $b = 0.1$.

it does not use a uniform random function. Instead, CHAT chooses the next destination using a chaotic system (Lozi map or Rössler system in this case). With the same trait as Random Walk, CHAT carries over the same features as Random Walk, while having its performance enhanced thanks to the chaotic system integration. The whole process of CHAT is shown in Alg. 1.

Algorithm 1: CHAT Process.

Algorithm: Main Algorithm

Data: α, Termination Condition, Graph, Strategy
Result: Discovery nodes List
Initialize initial conditions for Rössler system;
Initialize an Agent with a starting point;
Initiate `Lozi Map` (Initial condition, a, b);
Initiate `Rössler System` (Initial Condition, α);
while *termination condition is not met* **do**
    `Chaotic Traversal` (Agent, Strategy)
Return out the result;

Algorithm 2: Generating Lozi map solutions.

Algorithm: Lozi Map

Data: Initial conditions, a, b
Result: ρ (normalized solution)
Loop
    Simulate and solve Lozi map equations numerically;
    Normalize x_n to obtain ρ_n (Normalized solution);
    Return ρ_n;

There are two versions of our algorithms: *Vanilla version* and *Circular version*. The Vanilla version is a completely memoryless algorithm, while the Circular version, being an improved algorithm, requires a very small amount (almost negligible) of constant memory in order to enhance the coverage performance

Algorithm 3: Generating Rössler system discretized solutions.

Algorithm: Rössler System

Data: Initial conditions, α
Result: ρ (discretized solution)
Loop
    Simulate and solve Rössler system numerically;
    if *the solution is in the Poincaré Section Eq.* (4) **then**
        Normalize y_n to obtain ρ_n (Normalized discretized solution);
        Return ρ_n;

Algorithm 4: Agent Movement.

Algorithm: Chaotic Traversal

Data: Agent, Strategy
Get neighbor nodes from an agent position;
Assign equal priority to each neighbor;
Retrieve ρ from a Rössler system or a Lozi map;
if *Strategy == Circular* **then**
    Circulate the neighbor (Fig. 12)
Choose the next destination based on retrieved ρ;
Move to the destination;

of the Vanilla version. This section also presents the performance metrics used to compare the performance of CHAT and Random Walk in this work.

4.1. *Vanilla version*

In this version, the agent decides its next move based on the first return map values that the chaotic system gives out from Alg. 2 or Alg. 3. This version only utilizes the local information such as the list of neighbors and edges without storing any path history, making it completely memoryless. To elaborate how it works, the Fig. 11 is given and the whole process of movement is shown in Alg. 4. From the current node 1, the agent had four choices. For each choice, it was given the same priority as can be represented by this set; $Neighbors = (2 : 0.25, 3 : 0.25, 4 : 0.25, 5 : 0.25)$. As mentioned in section 3, the solution from the system is normalized to be in range $[0, 1]$. Therefore, if the system gives out 0.7814 after discretization, the agent would go to node 5. From there, the agent learns of the neighbors of node 5 and continues the process accordingly.

However, we have found one flaw in using the chaotic system for the graph traversal problem. In section 3, we mentioned that the solution from the chaotic system is aperiodic or in other words, never gives out two same values periodically. However, the first return map is a combination of patterns where the pattern itself might repeat. With a very low chances of happening, the agent might enter in a loop. Considering this situation, the agent makes a tour, and when it arrives at the starting point again, for the next iterations, the system gives out the same patterns. Hence, the agent repeat the same route all over again leading to failure in a graph discovery. We anticipate this problem and come up with a solution which incurs a negligible cost of memory.

4.2. *Circular version*

The word circular comes from the fact that we circulate the order of the neighbors in the list as in Alg. 4. In this Circular version, we still employ the chaotic system as a core function for choosing a next destination but add an ability to remember the latest node the agent has visited to prevent a loop traversal. The circulation goes by the deterministic rules. Still the advantage of circulation is not only to prevent the

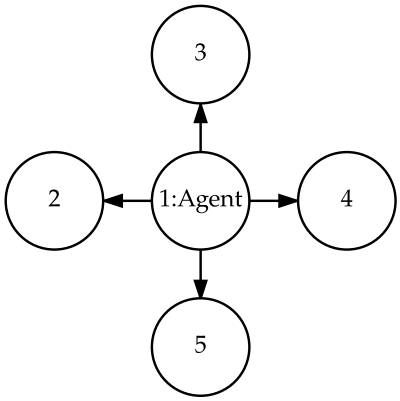


Fig. 11. Agent comes to the node 1 from node 4.

loop traversal, but it also improves the Coverage Percentage since we can place the previously visited node in the position that is not likely to be selected again so that the agent can avoid repetition. It can be explained using the Fig. 12.

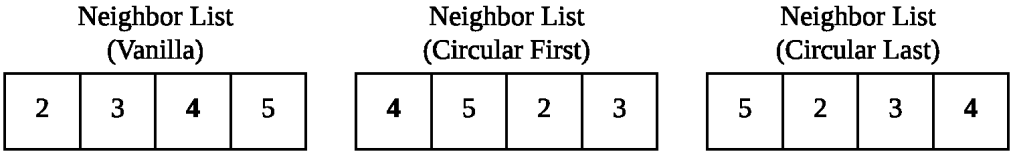


Fig. 12. The difference between Vanilla, Circular First and Circular Last neighbor list in the situation of Fig. 11.

From the Fig. 12, it shows the list of neighbors in the situation of Fig. 11. The length of the list equals to the degree of the node. From the figure, supposedly an agent comes to the node 1 from node 4 (in bold) and the current node has node 2, 3, 4 and 5 as its neighbors. In Vanilla version, the neighbors order is not reorganized, hence, the loop traversing can occur. It is different in the Circular version where the order is reorganized by shifting the previously visited node (in this case, node 4) to the first position in the neighbor list. Here, we propose two neighbor circulation methods, Circular First (as shown in Fig. 12) and Circular Last. The difference is the next position of the visited nodes: the Circular Last method left-shifts the previously visited node to the last position in the neighbor list. Combining these strategies with a first return map we can reduce the number of times that an agent revisits the already discovered nodes in a graph. Moreover, with this approach, even if the same pattern is to be revisited, the agent can still prevent itself from traversing in a loop.

4.3. *Performance Metric*

In this work, we propose two performance metrics, Coverage Percentage and Mean Time Coverage. The former is used to measure the exploration performance of an algorithm while the latter measures the exploration efficiency by considering discovered nodes in the graph and the algorithm execution time.

4.3.1. *Coverage Percentage*

We use the "Coverage Percentage" as a metric of exploration performance since typical performance metrics like "Traversal Time" which count how many iterations an algorithm takes to fully discover a whole graph cannot be used. This is due to the fact that real world graphs are unknown and the "Traversal Time" requires the size of a graph to terminate an algorithm and measure the performance. Therefore, "Coverage Percentage" is used instead since it reflects how many nodes an algorithm can discovered given a certain amount of algorithm iterations as shown in Eq. (12). For this metric, we let an agent run for n iterations

where n is a number of nodes in a graph. The chaotic system solving time and algorithm execution time are not considered since the chaotic solution sequences can be generated separately and the execution time is varied from machine to machine. The metric is defined as follows:

$$\text{Coverage Percentage} = \frac{\text{Number of Discovered Nodes}}{\text{Total Number of Nodes}} \times 100 \ . \tag{12}$$

4.3.2. *Mean Time Coverage*

This metric is used to measure the time efficiency of an algorithm. It calculates how many (different) nodes can be discovered in one second (CPU time). Regarding the execution, the algorithm can be viewed in two separated pieces, sequence generation (solving Lozi map and Rössler system or generate a random values sequence) and graph exploration. The first part can be prepared beforehand (offline) and does not affect the exploration part, which is considered to be an online process. Therefore, we only consider the CPU time required for the graph exploration. This metric is dependent on the hardware platform, but it can be used as a guideline for what to expect from each algorithm. The Mean Time Coverage is defined as follows:

$$\text{Mean Time Coverage (nodes/second)} = \frac{\text{Number of Discovered Nodes}}{\text{Graph Exploration CPU Time}} \ . \tag{13}$$

5. Experimental Results: Rössler CHAT

In this section, the experimental setup and the results are presented along with their discussions. Experimental results are divided in two parts. The first one contains the result of CHAT on a small graph in order to study the impact of α in the Rössler system. The second part presents the results on 1,000,000 nodes graphs to evaluate the performance of CHAT with Rössler dynamics on large graphs.

5.1. *Experimental Setup*

The test platform is implemented in Python (2.7) and NetworkX [Hagberg *et al.*, 2008]. The platform employs one agent in order to study the chaotic behavior. All tests are run on the High Performance Computing platform of the University of Luxembourg [Varrette *et al.*, 2014]. The algorithms are tested on five graph topologies as listed in Tab. 1. All tested graphs are undirected and unweighted graphs. For Ring, Small World and Random graphs, we generate them using Watt-Strogatz Generator. For Grid graphs, 2D-Grid Generator is used. Lastly, we use Power-Law Cluster Graph Generator to generate the Power-Law graphs. All the parameters being used are presented in Tab. 1.

All the graphs in the experiment are either small (2,000 nodes and 2,500 nodes) or large graphs (1,000,000 nodes). The 2,000 nodes graphs are used to show the impact of α on the Coverage Percentage of CHAT. We generate the sparse graph by giving the generator a low node degree. For each topology, there are 10 different graphs and each one are tested 30 times with Rössler system based CHAT for both small and large graphs with different starting points in each run. In case of grid topology, we employ an agent on several shapes, not just a square one. In addition, to obtain the final α, we extensively test each value of α and the effect of the performance on each topology to retrieve the ones that yield the best performance to test them on large graphs. Finally, the algorithms are compared against Random Walk (see Alg. 5) where the random values sequence can also be pre-computed offline like for chaotic systems. The reason for choosing Random Walk for comparison is because Vanilla CHAT and Random Walk are memoryless. In addition, the Circular CHAT, on the other hand, requires only marginal constant memory, while, other algorithms in the survey require memory to operate.

This section is organized into three main parts. The first part presents the result of the small graph experiment. The second part show the result of the large graph experiment and lastly, the study of α effect is presented in the last part.

Table 1. Experiments formulation. The best values of α obtained from the small graph experiments are used to explore large graphs. Every α is tested for 300 runs.

Topology	Small Graph Experiment	Large Graph Experiment
Ring	$n = 2000$, $k = 10$, $p = 0$	$n = 1000000$, $k = 10$, $p = 0$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 0.78$
Smalll World	$n = 2000$, $k = 10$, $p = 0.01$	$n = 1000000$, $k = 10$, $p = 0.05$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 0.78$
Random	$n = 2000$, $k = 10$, $p = 1$	$n = 1000000$, $k = 10$, $p = 1$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$
Grid	dimensions: 5×500, 10×250, 20×125, 25×100 and 50×50	dimensions: $50 \times 20000, 100 \times 10000, 160 \times 6250$, $200 \times 5000, 320 \times 3125, 400 \times 2500$, $500 \times 2000, 625 \times 1600, 800 \times 1250$ and 1000×1000 Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$
Power-Law	$n = 2000$, $m = 10$, $k = 0.2$	$n = 1000000$, $m = 10$, $k = 0.2$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$

Algorithm 5: Random Walk of an agent.

Algorithm: Random Walk

Data: Graph
Result: Discovery nodes List
Initialize an Agent with a starting point;
Generate random values sequence from a uniform random function;
while *termination condition is not met* **do**
    Retrieve a value from a sequence;
    Pick a destination based on a random value;
    Move to the chosen destination;
Return out the result;

5.2. *Results on Small Graph Instances*

In this section, we show the result of our algorithm comparing to Random Walk and also effects of varying the value of α in the Rössler system to analyze the influence of the dynamical properties on the behavior of our algorithms. From the results in this section, we are able to observe coverage differences of each approach and that aids us in choosing which combination of approach to be tested on larger graphs. For all figures in this section, the y-axis is the Coverage Percentage and the x-axis is the value of α supplement to the Rössler system. The black solid line is the result of the Circular First CHAT and the solid gray line is the result of Vanilla CHAT. Only Circular First method (named as "Circular" in all presented figures) is shown here due to its superior performance comparing to the Circular Last method. Lastly, the dashed line represents the result of the Random Walk. The shaded areas contain the result of the periodic solutions (in Rössler system) that does not exhibit chaotic dynamic behavior. Hence, we do not take them into consideration since the sole focus of our work is on addressing graph traversal problem with chaotic dynamics. Please also note that it is not possible to shade all the periodic regions due to the fact that some periodic parts are small and located in between chaotic regions. In this work, we consider the value of α in the range $[-0.5, 1.2]$ because the dynamical analysis is already performed for each attractor solution and from Fig. 4, values of α that are less than -0.5 mostly yield periodic attractors. We are thus able to compare the chaotic dynamics properties in this range of values.

5.2.1. *Ring Topology*

The nature of the ring topology is a structured graph. Referring to Fig. 13, it can be seen that when the system gives out the periodic solution, there is a sharp rise in the Coverage Percentage for the Circular version of CHAT. In fact, all the sharp peaks (above 20%) for the Circular version are periodic solutions, e.g., at $\alpha = -0.41$ and $\alpha = 1.0$. These periodic period are too small to shade. The average Coverage Percentage for the Circular version is about 10%. The highest Coverage Percentage (about 17%) for the Circular version is obtained with $\alpha = 0.78$. The Vanilla CHAT cover about 60%. While the Circular CHAT also has a good Coverage Percentage varied by the α. All in all, both chaos-based algorithms outperform Random Walk which only cover about 10% of the whole graph.

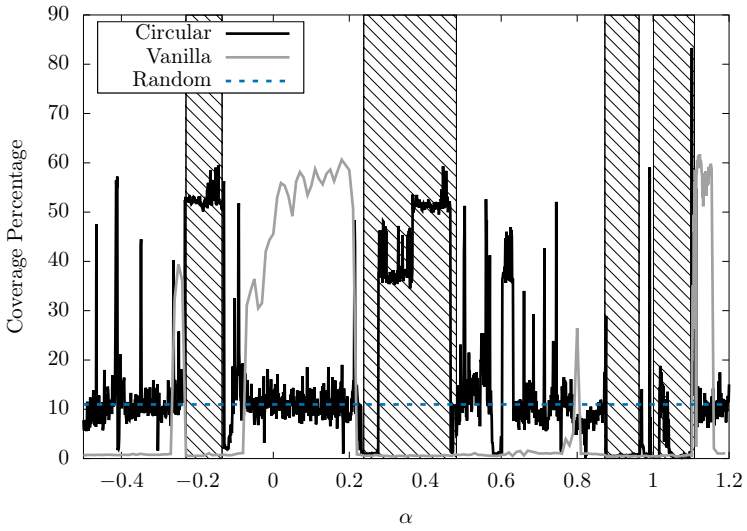


Fig. 13. Performance of Rössler CHATs in comparison to the Random Walk on the ring topology. The black solid line is the result of the Circular CHAT and the gray line is the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The shaded areas highlight periodic solutions of the Rössler system.

5.2.2. *Small World Topology*

This topology exhibits the theory of six degree of separation which is one of the most important aspects of the social network [Watts & Strogatz, 1998]. In the small world topology, Vanilla CHAT is able to outperform in Coverage Percentage of the Random Walk by 8% at $\alpha = 1.123$ with its 50% Coverage Percentage. The Circular version, even though, has lower Coverage Percentage than the former, is still able to produce a better performance than the Random Walk as well. However, when we compare Fig. 13 and Fig. 14, it seems like the effect of periodic solutions and chaotic solutions (in Rössler dynamics) become less distinctive as the graphs are less structured.

5.2.3. *Random Topology*

Random Graph is one of the most used topology to capture the essence of the real-world graphs [Lovász, 2012]. While, this topology should be the most well suited for the Random Walk due to its randomness, we can still see the increase of Coverage Percentage from the Circular CHAT over the Random Walk in Fig. 15. This is due to the circulation strategy that reduce the number of times an agent revisits already discovered nodes. An improvement of 4% in coverage over Random Walk is reported at $\alpha = 1.123$ with Circular version. The Coverage Percentage around $\alpha = 0.2$ can be disregarded since it falls in the periodic category. As for the reason why the Circular CHAT can obtain such a high Coverage Percentage, it is due to the understanding of the behavior of the system. By understanding the Rössler system, we can strategically place neighbors to avoid repetition. With a uniform random function, this is not possible due

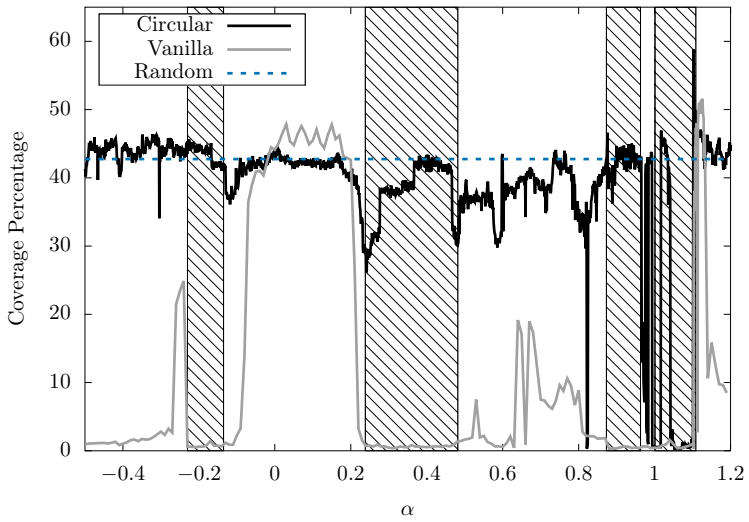


Fig. 14. Performance of Rössler CHATs in comparison to the Random Walk on the small world topology (see Fig. 13 definitions).

to its randomness.

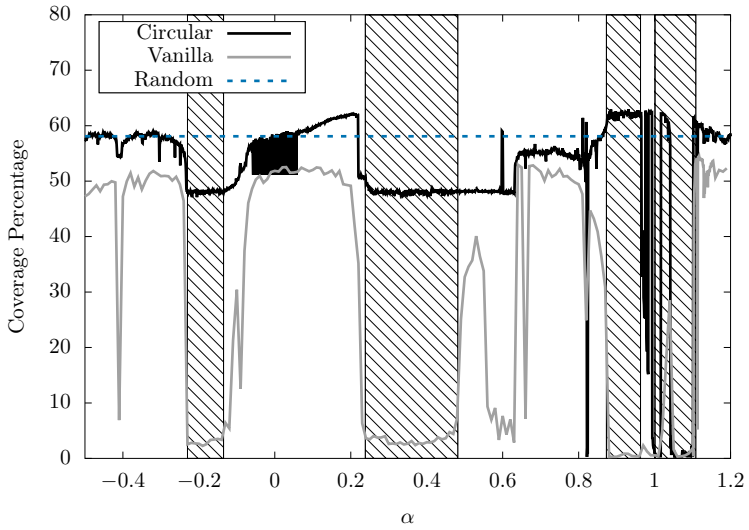


Fig. 15. Performance of Rössler CHATs in comparison to the Random Walk on the random topology (see Fig. 13 definitions).

5.2.4. *Grid Topology*

Even though, the grid topology is also a structured topology like ring topology, the achieved result is quite different from the ring graphs. The Vanilla CHAT Coverage Percentage is lower in grid topology compared to the ring topology. It is suspected that this is due to the effect of the locality of neighbor. Since the nodes are tightly connected together and share same neighbors, it is easier to send an agent into a loop traversal. Several peaks (e.g., at $\alpha = -0.41$, 0.6, and 0.87) are spotted from the Circular version. They can be disregarded as they fall in the periodic period. However, at $\alpha = 1.124$, not only the Vanilla CHAT obtain the highest Coverage Percentage, the Circular CHAT also yields the Coverage Percentage of 27%which is higher than the Random Walk as shown in Fig. 16 due to the ability to avoid the revisiting of the previous nodes. At this alpha, Circular CHAT shows 10% improvement in Coverage Percentage over Random Walk and over 18% compared to the Vanilla version.This piece of result emphasizes the need of an ability to

study the system to enhance the performance of the algorithm even further which is not available on a uniform random function.

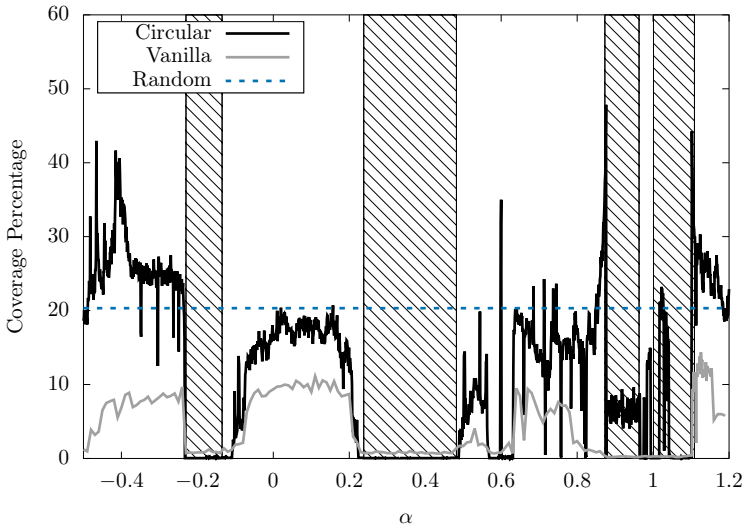


Fig. 16. Performance of Rössler CHATs in comparison to the Random Walk on the grid topology (see Fig. 13 definitions).

5.2.5. *Power-Law Topology*

Referring to Fig. 17, the Random Walk, and Vanilla version can cover a similar amount of nodes in the graph. On the other hand, the Circular CHAT shows a higher Coverage Percentage than other algorithms in this topology. A 2% improvement in Coverage Percentage at $\alpha = 1.124$. We also do not consider the Coverage Percentage around $\alpha = 0.2$ as the gaps in the solution make it behave like a periodic solution. It is worth to note also that the result observed from Fig. 17 is similar to the result of CHATs shown in Fig. 15. This is because both have similar randomness properties, but with different rules of constructing the graphs.

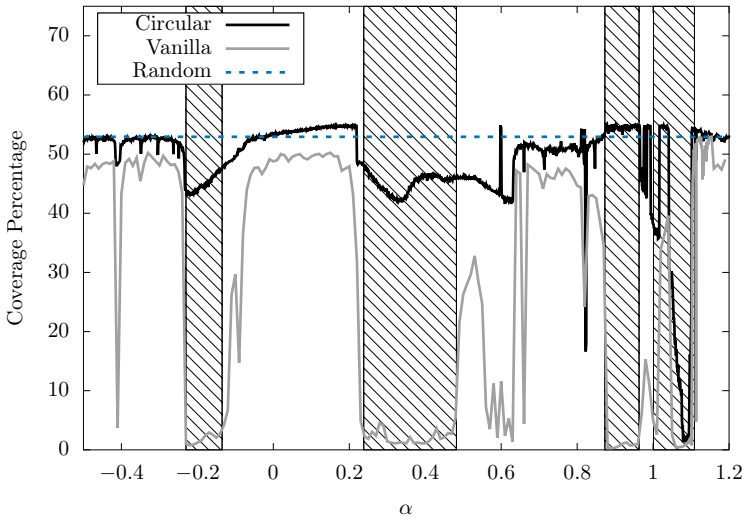


Fig. 17. Performance of Rössler CHATs in comparison to the Random Walk on power-law topology (see Fig. 13 definitions).

From all presented figures, it is clear that with a combination of periodic solutions and Vanilla version from the chaotic system, the Coverage Percentage is low. Yet, with the Circular version, we are able to

get a good Coverage Percentage on all topologies. This initial result on small graphs enables us to have a rough estimation of what α should be used in a larger experiment. From the algorithm point of view, the result should not change much since an agent in CHAT algorithm only takes a local information from the node in the graph and does not consider a big picture of the graph at all. Therefore, we test those values of α which give out chaotic solutions that yield a high Coverage Percentage in the larger graph experiment.

5.3. *Results on Large Graph Instances*

In this section, we present the best results for each algorithm on large graphs. The α being used in this experiment are taken from the top three α that yield the best Coverage Percentage for each topology and algorithm. Tab. 2 compare the results between CHAT (both versions) and Random Walk using Kruskal-Wallis test [Kruskal & Wallis, 1952]. In this table, the result is expressed in a form of Coverage Percentage. Furthermore, the comparison between each algorithm on each topology is conducted using Wilcoxon test [Wilcoxon, 1945] in Tab. 3. Both Kruskal-Wallis and Wilcoxon tests are known for testing the difference in means of targeted populations. Kruskal-Wallis test is normally used when there are more than two populations to test, while Wilcoxon mainly test between two populations. For both tests, we use a confidence interval of 95%.

Table 2. Coverage Percentage of CHAT and Random Walk. The first number is the Coverage Percentage. The second number behind ± is the standard deviation.

Topology	Random Walk	CHAT (Rössler Vanilla)	CHAT (Rössler Circular)	$p-$value
Ring	0.521 ± 0.155	**60.471 ± 0.001**	0.5645 ± 0.002	< 2.2e-16
Small World	42.219 ± 0.101	**50.504 ± 0.087**	45.956 ± 0.112	< 2.2e-16
Random	58.101 ± 0.003	52.761 ± 0.061	**60.218 ± 0.001**	< 2.2e-16
Grid	14.419 ± 3.734	0.690 ± 0.606	**18.052 ± 4.108**	< 2.2e-16
Power-Law	51.236 ± 0.032	51.195 ± 0.035	**54.045 ± 0.011**	< 2.2e-16

From Tab. 2, the result show that CHATs can perform significantly better than traditional Random Walk in all tested topologies. In ring topology, we are able to see an improvement of 60% from Vanilla CHAT with the Rössler dynamic over Random Walk, while Circular version yields relatively the same Coverage Percentage as Random Walk. The next topology is small world topology where CHAT (Vanilla version with Rössler dynamic) having the highest Coverage Percentage, outperforms Random Walk by 8% in coverage and approximately, 2% of improvement in coverage over Random Walk is reported from Circular CHAT. Even with random topology, the significant improvement in coverage of 2% can be found and here Circular CHAT demonstrates that with a negligible constant memory, it yields the best Coverage Percentage. As in grid topology, the Circular version outperforms the Random Walk in term of Coverage Percentage significantly by 4%. As for power-law topology, CHAT (Circular version) still displays a significant improvement of 3% in coverage over Random Walk, while the Vanilla version shows a similar result to the random Walk.

It is clearly shown in the table that CHAT dominates the Random Walk in terms of Coverage Percentage in all tested topologies. This result also shows that they still follow the same trend even if there are some discrepancies of the Coverage Percentage which can be accounted by the random starting points, a difference in size and dimensions, plus the randomness during graphs construction. For example, from Fig. 13, Circular CHAT performs better than the Random Walk, and in large graphs, Circular CHAT still perform better. Even though the improvement is small, but it is still proved to be significant. Moreover, in the grid topology, the presented result in Tab. 2 and Fig. 16 are also similar, although, there is a drop in Coverage Percentage from a small experiment in a large graphs. However, that is to be expected given that more different grid dimensions are used to test the algorithm.

To further analyze the result, the matrix of Wilcoxon test is shown in Tab. 3. The Wilcoxon test indicates the significant difference in the means of two populations. We thus can determine which algorithm performs better by the statistical significance and their differences in Coverage Percentage. The upward

Table 3. Wilcoxon test between each algorithm where ▲, ▼ mean that the result is better, respectively worse, significantly The five symbols in a column represent a performance for each topology: Ring, Small World, Random, Grid and Power-Law respectively

Algorithm	Random Walk	CHAT (Rössler Vanilla)	CHAT (Circular)
Random Walk		▼▼▲▲▲	▼▼▼▼▼
CHAT (Vanilla)	▲▲▼▼▼		▲▲▼▼▼
CHAT (Circular)	▲▲▲▲▲	▼▼▲▲▲	

triangles mean that the result is better significantly and the downward ones, otherwise. The circle represents that there is no statistical significance in the difference between each algorithm for a certain topology, hence two algorithms yield similar Coverage Percentage for a certain topology. The five symbols in a column represent a performance for each topology, ring, small world, random, grid and power-law respectively. From the Tab. 3, it can be observed that the Circular version of CHAT is better than Random Walk in all tested topologies and perform significantly better than the Vanilla version in random, grid and power-law topologies. From these results, we suggest that Vanilla Rössler CHAT should be used for ring and small world topologies, while Circular version should be used for grid, random and power-law topologies. From both small and large graph experiments, we gather all the result and conduct an in-depth analysis on α. The analysis concerns both the effect of α and algorithm optimization.

5.4. *Study of Chaotic Dynamics Impact*

In this section, the analysis of performance for both periodic and chaotic solutions are presented. We use first return maps as an analysis tools to explain the system behaviors and how they affect CHAT performance. From the values obtained from the previous figures, we can obtain the first return maps associated to each value of α. The example of the first return maps for periodic solutions is shown in Fig. 18 which is the solution given by the system when $\alpha = 0.33$. From this first return map, it is self-explanatory as why it is periodic since there are only a few points on this map. In contrast, Fig. 19 and Fig. 22 display first return maps of the chaotic solution. The difference between the three figures is quite obvious since in the first return maps of chaotic solution, there are more points than the periodic one which means more possible values can be obtained from the system unlike, in Fig. 18 where only three points are presented. In the following sections, we discuss the effect of periodic and chaotic solutions on CHAT.

5.4.1. *Effect of Periodic Solutions on CHAT*

It can be seen in Fig. 13 and Fig. 16 when the system gives out periodic solutions, Circular CHAT performs really well and even better than the Random Walk. In fact, for a structured graph, there is not a need for randomness at all. The structured graphs can be traversed efficiently with simple systematic rules and a path history to prevent it from traveling in a circle. Therefore, this is not in the essence of our work since we are interested in the chaotic behavior and aims at real world graphs which are unstructured. Despite that, it is still important to describe this event to explain the cause of sharp rise in those result figures. As shown in Fig. 18, given only a few possible choices to move should not result in a high Coverage Percentage and that is true as shown in every tested topology with the Vanilla CHAT. From the analysis, the agent moves in circle due to limited choices in destination as shown in Fig. 21 and this can be seen in all small experiment figures. However, in Circular CHAT, by circulating the neighbors list, the algorithm becomes more like a port selection technique.

A port selection technique [Ilcinkas, 2008] is a traversal technique in a network. In a network graph, a server is considered to be a node and each connected port on a server is an edge in the graph. From the starting point in a graph, the agent chooses to traverse on the first edge (the first connected port on the list) and that edge (or port) is then marked. On every node, the agent chooses the next unmarked edge (the next connected port on the list) to be traversed on and once the agent visits a node that has all edges marked, the marks are removed, and the same process starts over. The Circular CHAT and a periodic solution is similar to this technique. The circulation of neighbors in the list is similar to marking the edge

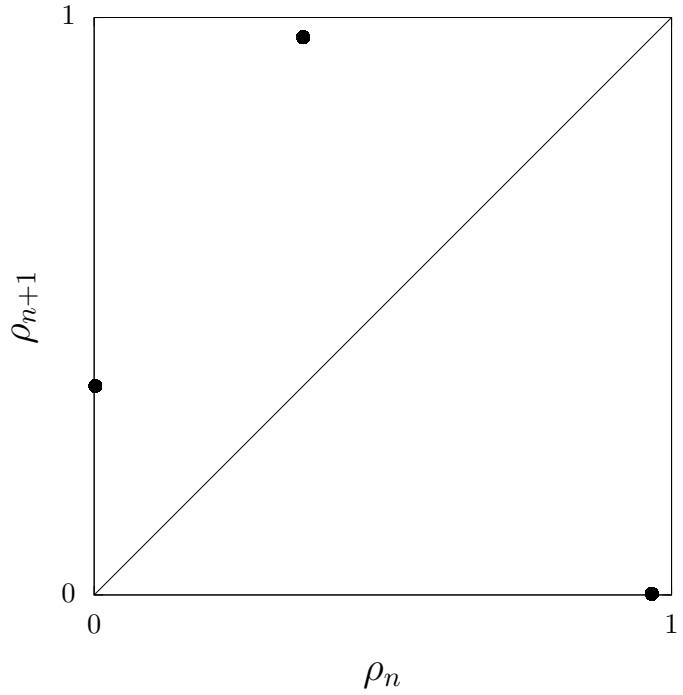


Fig. 18. First Return Map to the Poincaré section for $\alpha = 0.33$.

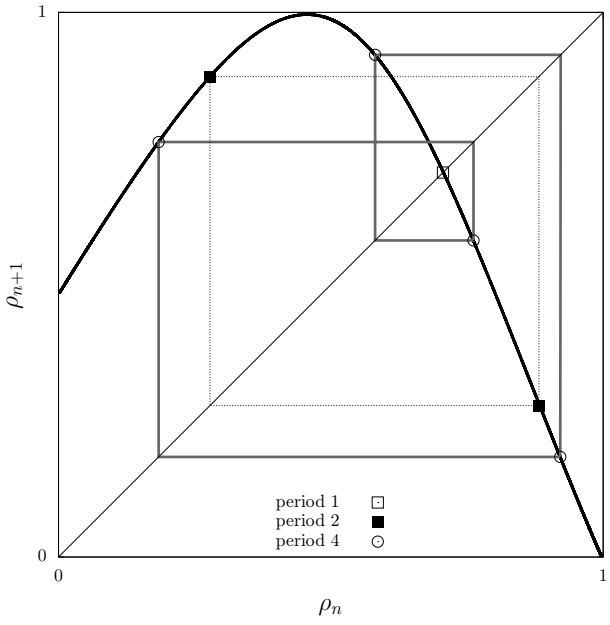


Fig. 19. First Return Map to the Poincaré section for $\alpha = -0.26$. Periodic points are indicated to underline the structure of the first return map. These periodic points corresponds to orbits in the attractor. Only periodic points associated to orbits with a period lower than five are indicated.

and with few possible values from the periodic solution, we can move the previous nodes to the position that will not be selected. For example, there exists a period-1 solution. In that case, we can strategically move the previous node to a position that will not be selected and promote the unvisited node to be chosen. Thus, the agent has less chance to visit the already visited nodes and higher chance to explore the graph. As in Fig. 18 which is a period-3 solution, it produces a higher Coverage Percentage in Grid and Ring topologies because these two topologies are structured and this port selection technique happens to be very

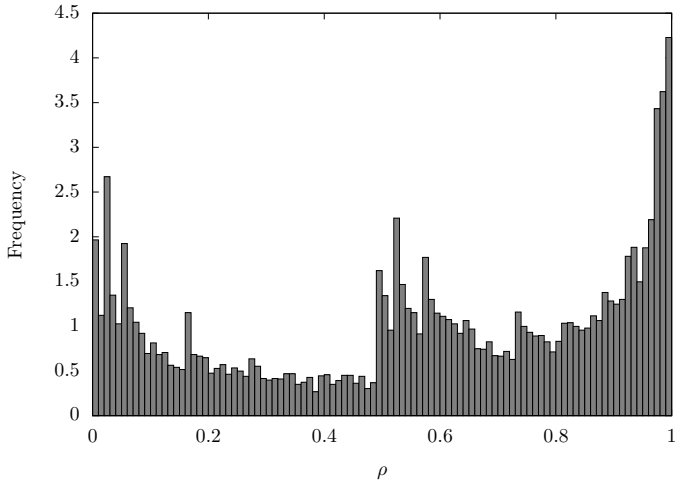


Fig. 20. The density of iterates at $\alpha = -0.26$.

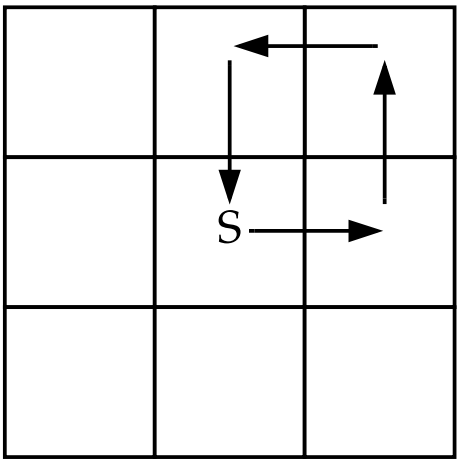


Fig. 21. Periodic solution and traversal of an agent in a grid from the starting point (S).

efficient in a structured graph traversal. With this, we explain the cause of high Coverage Percentage in periodic regions presented in figures in Section 5.2. Next, we discuss on our main focus of this work which is how chaotic dynamics affect CHAT performance.

5.4.2. *Effect of Chaotic Solutions on CHAT*

To study the effect of chaotic dynamics thoroughly, the focus is on the Vanilla CHAT. This is because there is no modification being done on the Vanilla CHAT, hence it exhibits a pure chaotic behavior. In all tested topologies, the periodic solutions fail to cover the graph efficiently compared to the chaotic solutions. To elaborate the cause of this result, considering a maze walking problem, there is a right hand rule where the agent always walk on the right side of the wall. With this rule, one is able to exit the maze. However, exploration is completely different. If only one rule is to be followed without any regards of the previous path. There is a chance that an agent will be trapped at some point. Hence, if only one rule is applied to traverse the graph, it would be the same as Breadth First Search (BFS) or Depth First Search (DFS). Coupling with the issue of no memory of the traversing path, an agent ends up being caught in a circle at some point which is the reason why, the relocation function is needed in BFS and DFS for traversing an unknown graph [Albers & Henzinger, 2000]. In Vanilla version, the neighbors list is deterministic. The order of neighbors of each node does not change over time. If we take the Fig. 18 as an example, there are

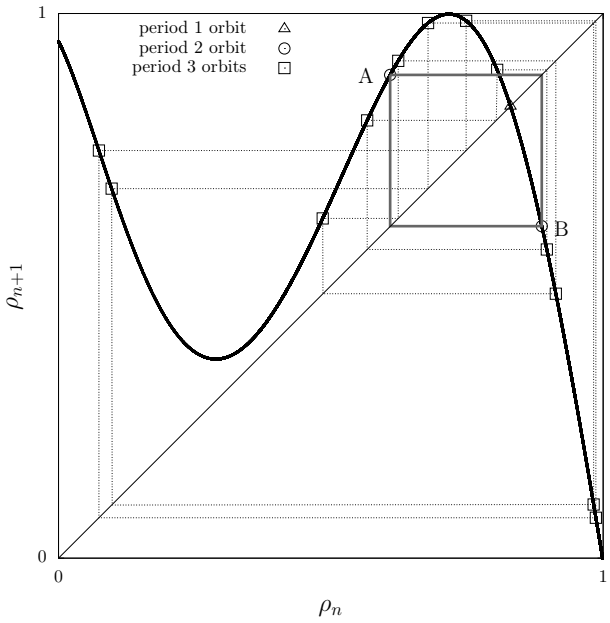


Fig. 22. First Return Map to the Poincaré section for $\alpha = 1.124$. Periodic points represent periodic orbits in the three-dimensional space. Period 2 orbit is thus represented by the points $A = (x_a, y_a)$ and $B = (y_a, x_a)$ in the first return map. Only periodic points associated to orbits with a period lower than four are indicated.

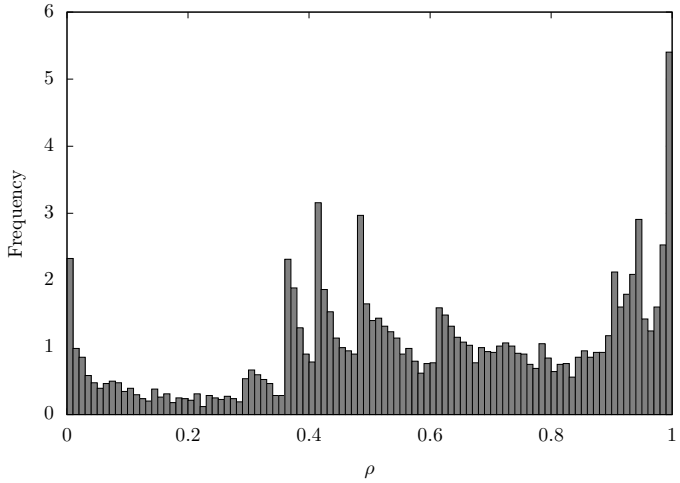


Fig. 23. The density of iterates at $\alpha = 1.124$.

only three points on the map. This is similar to having only three rules. Thus, at the fourth iteration, if the agent happens to revisit the starting node again, the same route will be repeated over. The chaotic solutions are different. In Fig. 19 and Fig. 22, there are more points on those two maps compared to Fig. 18. Each point on the map can be considered as a rule for the agent. With the chaotic solution, for a specific amount of time, the agent is given a set of rule (in this case, a pattern from the system) to traverse the area, but this rule changes over time to offer a breakthrough from potential circle without a need for a memory of the path. It can be further elaborated with the orbits in Fig. 19 and Fig. 22. There are periodic points in these two first return map that are temporarily reached and corresponds to orbits. These orbits are unstable, thus the periodic solution do not last. We only provide periodic points for low period orbit to detail the first return map dynamic. Low period orbits are more often visited. But it can be observed that there are more points in the orbits in Fig. 22 which means more rules are available. In addition, the density of iterates (further read in [Hamaizia *et al.*, 2012]) are shown in Fig. 20 and Fig. 23. These two figures depict the density of values in the first return maps. It can be observed that there are more

distinguished peaks in Fig. 23 than in Fig. 20. This difference shows that there are around four prominent choices (frequency above 1) if $\alpha = -0.26$ is used in CHAT. On the other hand, there are seven peaks in the first return map of $\alpha = 1.124$. This is the reason why we see a better Coverage Percentage from the chaotic solution over the periodic solution in Vanilla CHAT on all tested topologies and also explain why a more chaotic behavior can perform better than a standard one.

As the results are shown Figs. 13 – 17, it is clear that chaotic behavior is indeed the key to higher Coverage Percentage on the graphs for CHAT. However, even chaotic solutions are different as it can be observed that for different values of α. This is due to the complexity of the chaotic behaviors. In Fig. 19, there are two branches. From $\rho_n = [0, 0.45]$, this is the first branch and it is increasing. The second branch is from $\rho_n = (0.45, 1]$. While in Fig. 22, there are three branches. The first branch is a decreasing branch starting from $\rho_n = [0, 0.3]$. The second branch is from $\rho_n = (0.3, 0.7]$ and it is an increasing branch. The last branch is a decreasing branch and starting from $\rho_n = (0.7, 1]$. The higher number of branches in the first return map, the more complex the chaotic behavior. Considering Fig. 19 and its given value at each iteration, the consisted patterns do not change abruptly. Some patterns even continue for a number of iterations as shown in Fig. 24. This is the same problem with the periodic solution. If the agent arrives at the node where the pattern first starts, it is bounded to continuously move in a circle until the next pattern comes. Therefore, with a more complex chaotic behavior in the banded chaos area where the first return maps contain more than two branches, more patterns exist and can be changed abruptly leading to a wider coverage on a graph as can be seen in the result figures of each topology where $\alpha = 1.123$ and $\alpha = 1.124$ dominate other values of α in Coverage Percentage. When this complex chaotic behavior is combined with the Circular strategy, we are able to gain even better Coverage Percentage since we can finally avoid repetitions that come with patterns in chaotic dynamics.

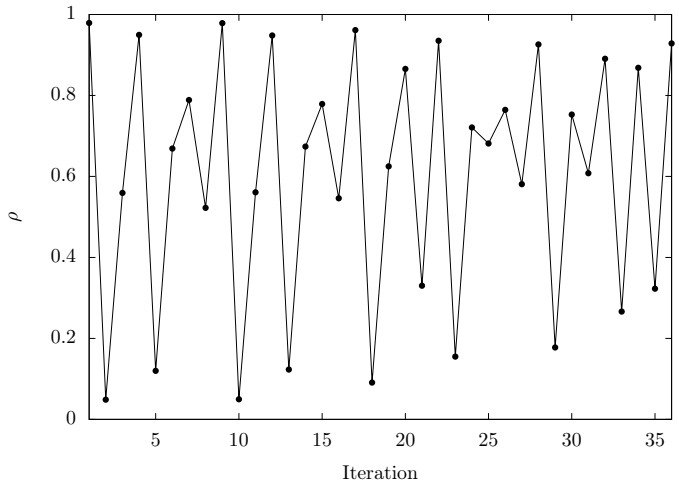


Fig. 24. A plot of ρ against iteration.

6. Experimental Results: Lozi CHAT

This section presents the results of CHAT with the Lozi map and its comparison to CHAT with the Rössler system which showed superior performance than Random Walk in the previous section. This section contains three main parts. The first one presents the experimental setup. Then, the preliminary results on small graphs are presented in the second part. In the last part, the results on large graphs are presented together with their discussion.

6.1. *Experimental Setup*

The same test platform and libraries have been used to test CHAT with the Lozi map to ensure a fair comparison (cf. Section 5.1). The Lozi map is based on two parameters, a and b. These two parameters

Table 4. Experiments formulation. The best values of a and b obtained from the small graph experiments are used to explore large graphs. Selected combinations of a, b, and method are tested for 30 runs on each graph.

Topology	Small Graph Experiment	Large Graph Experiment
Ring	$n = 2000$, $k = 10$, $p = 0$	$n = 1000000$, $k = 10$, $p = 0$ $a = 1.5$, $b = -0.44$ Vanilla
Small World	$n = 2000$, $k = 10$, $p = 0.01$	$n = 1000000$, $k = 10$, $p = 0.01$ $a = 1.5$, $b = -0.42$ Vanilla
Random	$n = 2000$, $k = 10$, $p = 1$	$n = 1000000$, $k = 10$, $p = 1$ $a = 1.78$, $b = 0.1$ Circular Last
Grid	dimensions: 5×500, 10×250, 20×125, 25×100 and 50×50	dimensions: $50 \times 20000, 100 \times 10000, 160 \times 6250$, $200 \times 5000, 320 \times 3125, 400 \times 2500$, $500 \times 2000, 625 \times 1600, 800 \times 1250$ and 1000×1000 $a = 1.43$, $b = 0.1$ Circular Last
Power-Law	$n = 2000$, $m = 10$, $k = 0.2$	$n = 1000000$, $m = 10$, $k = 0.2$ $a = 1.5$, $b = 0.41$ Circular Last

allow many variations, however it is not the main goal of in this work to explore them all. Therefore, two sets of parameters are used in these experiments, $a = 1.5, b = [-0.5, 0.47]$ and $a = [1.1 - 1.8], b = 0.1$ which were previously studied in [Botella-Soler *et al.*, 2011]. We also choose to use the value of x_n scaled to $[0, 1]$ which is the same approach as [Araujo & Coelho, 2008]. Each algorithm is tested on each graph for 30 runs with different starting points. The experimental configuration is shown in Table 4 where the parameters for the large graph experiments are chosen from the configurations that yield the highest Coverage Percentage in the small graph experiment.

6.2. *Results on Small Graphs Instances*

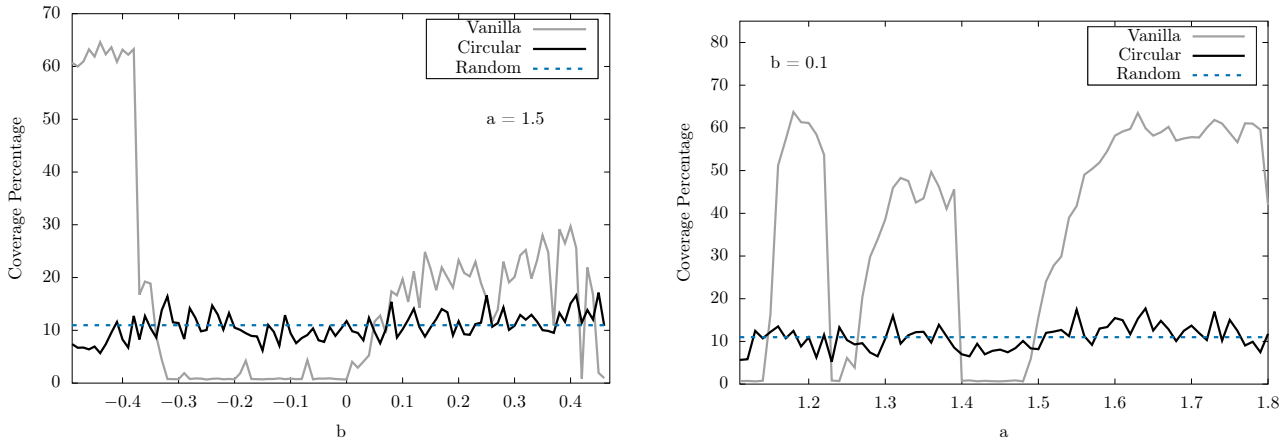
The results in this section serve as a screening process and help us in obtaining the best combination of parameters and algorithms for the large graphs experiments. All the result figures in this section present the Coverage Percentage on the $y - axis$. The x-axis represents either a or b. The obtained solutions from the Lozi map with the selected parameters are all chaotic, hence there is no shaded area in the figures. In all presented figures in this section, the word "Circular" refers to the Circular Last method. For simplicity, the Circular First method is not shown due to its inferior results.

6.2.1. *Ring Topology*

Both versions of Lozi CHAT are reported to have a superior Coverage Percentage than the Random Walk as shown in Fig. 25. The Vanilla version can achieve around 60% Coverage Percentage which is 50% higher than Random Walk. This result is also similar to the Rössler CHAT where the Vanilla version performs best on the ring topology. From the results, we selected the Vanilla method with parameters $a = 1.5$ and $b = -0.44$ to be used in the large graphs experiment since it provided the highest Coverage Percentage with almost 63%.

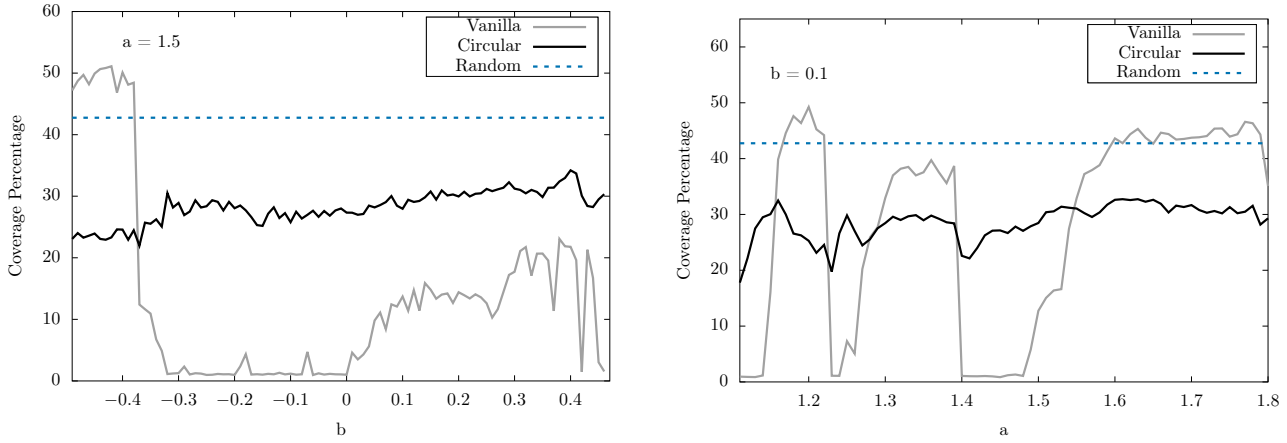
6.2.2. *Small World Topology*

On the small world topology, the Vanilla version also yields the highest Coverage Percentage at around 50% as shown in Fig. 26. The Circular version, on the other hand, yields lower Coverage Percentage than the Vanilla version and the Random Walk. From this result, the Lozi map also shares the same trait with the Rössler system in that the Coverage Percentage decreases when the graph becomes less structured. The combination of $a = 1.5$ and $b = -0.42$ is selected to be used in the large graph experiment due to its superior performance.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$. (b) $a = [1.1, 1.8]$ and $b = 0.1$.

Fig. 25. Performance of Lozi CHAT: both are compared to the Random Walk on the ring topology. The black solid line represents the results of the Circular (Last) CHAT and the gray line the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The Circular First method is not shown due to its inferior results compared to the Circular Last method.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$. (b) $a = [1.1, 1.8]$ and $b = 0.1$.

Fig. 26. Performance of Lozi CHAT in comparison to the Random Walk on small world topology (see Fig. 25 definitions).

6.2.3. *Random Topology*

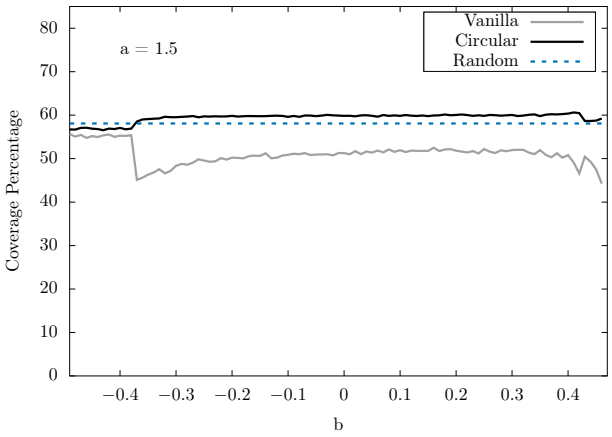
Fig. 27 shows that the Circular version of Lozi CHAT can achieve a higher Coverage Percentage than the Random Walk with a 2% improvement. In contrast, the Vanilla version performs worse than both with the peak Coverage Percentage at 56% only. With this result, we selected $a = 1.78$ and $b = 0.1$ for the large graph experiment for the random topology.

6.2.4. *Grid Topology*

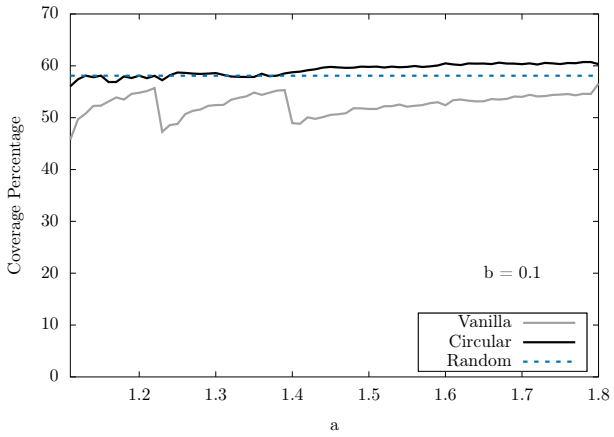
In the grid topology, the Random Walk obtains 20% coverage on average which is better than the Vanilla Lozi CHAT (refer to Fig. 28) that yields around 14% at most. On the other hand, the Circular version outperforms the Random Walk by over 8% in coverage at $a = 1.43$ and $b = 0.1$, hence these parameters are selected.

6.2.5. *Power-Law Topology*

The performance trend in Fig. 29 is similar to the trend in the random topology (Fig. 27). Even though, the Coverage Percentage of the Random Walk is quite comparable to the Circular version, at $a = 1.5$ and

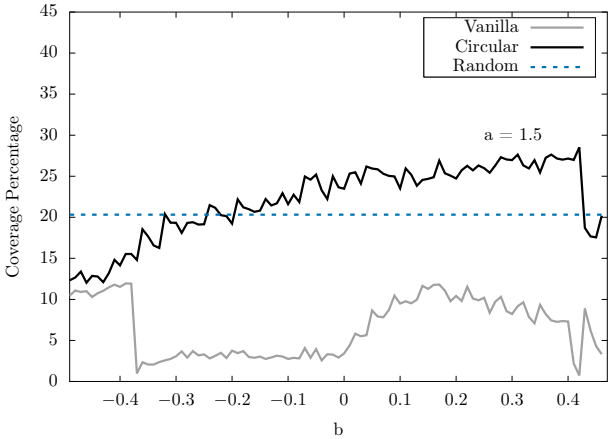


(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

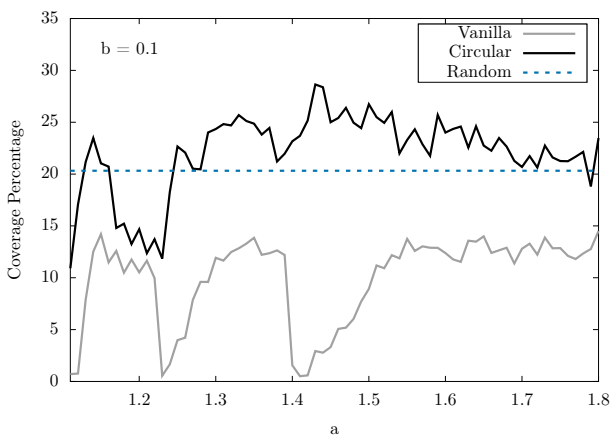


(b) $a = [1.1, 1.8]$ and $b = 0.1$.

Fig. 27. Performance of Lozi CHATs in comparison to the Random Walk on random topology (see Fig. 25 definitions).



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.



(b) $a = [1.1, 1.8]$ and $b = 0.1$.

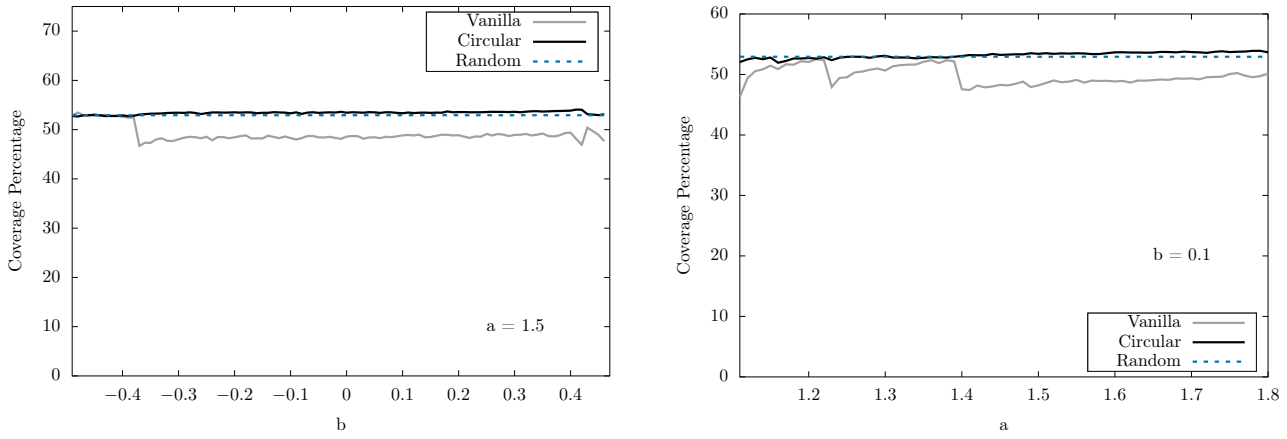
Fig. 28. Performance of Lozi CHATs in comparison to the Random Walk on grid topology (see Fig. 25 definitions).

$b = 0.41$, a 2% improvement in Coverage Percentage from the Circular version can be observed. From all these results, it seems that once the graph becomes structureless, the importance of Vanilla version for Lozi CHAT is diminished as it cannot be compared with the Random Walk.

6.3. *Results on Large Graphs Instances*

In this section, we use the selected methods and parameters that yield the highest Coverage Percentage from each topology and test them on a $1,000,000$ nodes graph. This graph is the same one that we use to test the Rössler CHAT in Section 5.3. In that section, it has also been concluded that Rössler CHAT yields a higher Coverage Percentage than the Random Walk for all topologies, therefore, we only compare the Coverage Percentage between Rössler and Lozi CHAT as shown in Tab. 5 using Wilcoxon test [Wilcoxon, 1945].

From Tab. 5, Lozi CHAT is significantly better than Rössler CHAT on the ring and random topologies. Lozi CHAT yields 2% more Coverage Percentage on the ring topology, while an improvement on the random topology is minor. As for Rössler CHAT, it outperforms Lozi CHAT on small world and power-law topologies, by almost 1% in Coverage Percentage. On the random topology, the difference in Coverage Percentage is marginal. Lastly, on the grid topology, both Rössler and Lozi CHAT are comparable in terms of Coverage Percentage. The standard deviation might be large, but this trend also persists in the Random Walk as well, due to the variation of the shape of the tested grids.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

Fig. 29. Performance of Lozi CHATs in comparison to the Random Walk on power-law topology (see Fig. 25 definitions).

Table 5. Best performance of Rössler CHAT and Lozi CHAT for each graph topology. The first number is the Coverage Percentage and the number behind ± is the standard deviation. For Rössler and Lozi CHAT configurations, please refer to Tab. 1 and Tab. 4 respectively.

Topology	Best Rössler CHAT	Best Lozi CHAT	$p-$value
Ring	60.471 ± 0.001	**62.475 ± 0.001**	< 2.2e-16
Small World	**50.504 ± 0.087**	49.739 ± 0.134	< 2.2e-16
Random	60.218 ± 0.001	**60.558 ± 0.034**	< 2.2e-16
Grid	18.052 ± 4.108	18.892 ± 6.548	0.8704
Power-Law	**54.045 ± 0.011**	53.344 ± 0.036	< 2.2e-16

6.4. *Study of Chaotic Dynamics Impact*

From the study of the Rössler system in Section. 5.4.2, we observe the effect of density of iterates in the Coverage Percentage. The solutions that have high fluctuation in ρ tend to yield higher Coverage Percentage than the ones that do not. Therefore, we study the density of iterates for $a = 1.5$, $b = -0.44$ and $a = 1.78$, $b = 0.1$ as they yield the highest Coverage Percentage for different versions of CHAT. The x-axis in the plot is ρ or normalized solution from the map and the y-axis is the frequency of the ρ. In Fig 30, the density of iterates for $a = 1.5$ and $b = -0.44$ is shown. This set of parameters yield a high Coverage Percentage in ring and small world topology. From the plot, there are many peaks above one (in frequency) which means an agent has more variations for choosing the destination. However, these peaks are close to each other and sometimes represent the same choice which is different from the Rössler system with $\alpha = 1.124$ in Fig .23 that has fewer peaks and these peaks are not clustered together leading to the higher variations in choices.

Another example is the density plot from the Lozi map with $a = 1.78$ and $b = 0.1$. This set of parameter works really well with Circular Last method. The reason becomes clear with Fig. 31. The majority of ρ is in $[0, 0.8]$ as many peaks can be observed in that area. This means when we shift the previous node to the end of the neighbor list, it has less chance to be visited again. With this result, the importance of the Circular approach is emphasized and it demonstrates that chaotic systems provide more promising results for graph exploration than the Random Walk.

From all the CHAT results, it shows that CHAT can outperform the Random Walk in terms of node discovery (Coverage Percentage). Based on the results in this section and the previous section, Lozi map should be used when traversing the ring and random graphs. While Rössler system should be used to traverse small world and power-law graphs. As for grid, both can be used, but Rössler system might be a better choice due to the lower standard deviation. The next section proposes to explore further the

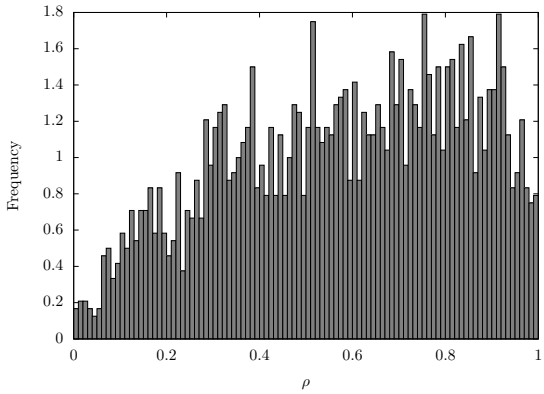


Fig. 30. The density of iterates of Lozi map for parameter values $a = 1.5$ and $b = -0.44$.

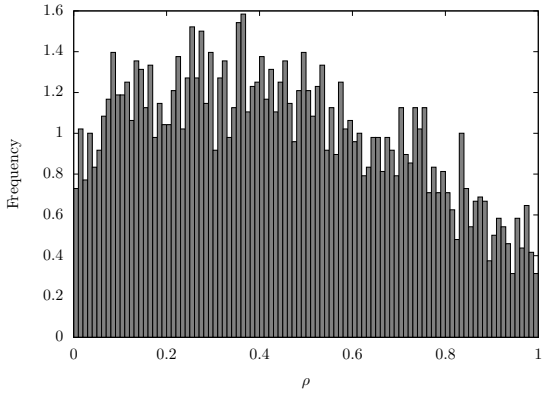


Fig. 31. The density of iterates of Lozi map for parameters values $a = 1.78$ and $b = 0.1$.

experimental results in terms of Mean Time Coverage.

7. Experimental Results: Mean Time Coverage

In this section, we report the results in terms of Mean Time Coverage (Eq. (13)) which is the ratio of number of discovered nodes over the algorithm's execution time. In CHAT and Random Walk, there are fundamentally two processes. The first one is to prepare the sequence of values for graph traversal (ρ in CHAT and random number in a range of [0,1] in Random Walk). The second part is to execute the traversal algorithm. For the Mean Time Coverage metric in this work, we focus on the second part where the algorithm is executed instead of the first part due to the fact that the sequences can be pre-computed (i.e., generated offline) and have thus no effect on the second part. However, we also report the sequence generation time in this section to complete the whole study.

7.1. *Test Platform*

Our implementation is in Python 2.7. The essential libraries are NetworkX 1.11, numpy 1.11.1 (for pseudo-random number generator (PRNG) with Marsenne Twister algorithm [Matsumoto & Nishimura, 1998]) and scipy 0.17.1. The test is performed on a single core of an Intel Xeon L5640 (2.26 GHz) with 4 GB memory available from the High Performance Computing (HPC) platform of the University of Luxembourg [Varrette *et al.*, 2014]. Every algorithm (with reported parameters that yield the best result) is tested on five topologies with the small graphs as shown in the small graph experiment setup in Tab.4 for 30 runs on each graph. The measured CPU time for online exploration and number of discovered nodes is the average of 30 runs. To accommodate with 2,500 node graph, we also measure the CPU time for generating a 2,500-value and 1,000,000-values sequences for Rösssler system, Lozi map and uniform random function (using the numpy library). The average CPU time for a sequence generation process is calculated from 30 runs as well. We select $\alpha = 1.124$ and $a = 1.5$, $b = -0.42$ as representatives for generating the ρ sequences for

Rössler system and Lozi map respectively since the computation time for these parameters is the highest (upper bound) in their respective systems.

7.2. *Results on Algorithm Computational Performance*

The Mean Time Coverage for each algorithm is shown in Tab. 6. These numbers reflect the number of nodes that can be discovered in one second by the algorithms. The Vanilla Lozi CHAT obtains the best result on the ring topology. On the small world topology, Vanilla Rössler CHAT yields the highest value with 8388 nodes in one second. The Circular First Rössler then dominates other algorithms on all other topologies. These values are compliant with the previous results (Tab. 3) since those algorithms provide a higher Coverage Percentage. However the outcomes are different on the random and grid topologies. It is shown in Tab. 5 that Circular Last Lozi CHAT yields a higher Coverage Percentage on the random topology and there is no significance in the difference between the latter and the Circular First Rössler CHAT on the grid topology. Yet, the Mean Time Coverage suggests otherwise. This is due to the execution time of the Circular Last methods which is longer than its counterpart leading to a lower efficiency. However, in any case, the efficiency of CHAT is higher than Random Walk on all tested topologies.

Table 6. Mean Time Coverage for each Algorithm in node/second. The CPU time and number of discovered nodes are calculated from 30 runs. The best results are in bold.

Algorithm	Ring	Small World	Random	Grid	Power-Law
Vanilla Rössler	10304	**8388**	8881	2447	6038
Circular First Rössler	2833	7565	**9677**	**5319**	**6625**
Circular Last Rössler	2343	7120	6645	2430	5306
Vanilla Lozi	**10641**	8299	9265	2534	6050
Circular First Lozi	8500	7000	9193	3191	6500
Circular Last Lozi	2656	5379	9375	4861	5510
Random Walk	1854	7077	9604	3602	6025

Considering CPU time, the exploration of the graph with our algorithm is quite comparable, however the bottleneck lies in the sequence generation time. The CPU time for generating the value sequences is shown in Tab. 7. It can be observed that the Rössler system takes the longest time to compute with around 20 seconds due to the solving of "Ordinary Differential Equation" (ODE) and "Runge-Kutta method (fourth order)" (RK4). The Lozi map is much faster than the Rössler system and is comparable to Mersenne Twister algorithm. As for the chaotic dynamics computation of Rössler system, one possibility to optimize further the computation time is to consider an FPGA implementation [Koyuncu *et al.*, 2014]. However this is not the objective of this paper and will be considered for future work. Regardless of the computing time for each system, the sequence can always be pre-computed, and hence, can be disregarded in the real scenarios. The reported results in this work indicate that CHAT can perform better than the Random Walk in all tested topologies in terms of Coverage Percentage and Mean Time Coverage. The Lozi map is also more suitable to use with CHAT than the Rössler system due to the significantly lower computation time to obtain discrete chaotic dynamics (number sequence generation).

Table 7. CPU time for generating the value sequence for Rössler system ($\alpha = 1.124$), Lozi map ($a = 1.5$, $b = -0.44$), and Marsenne Twister algorithm (in second). These CPU times are averaged from 30 runs.

Chaotic dynamics or PRNG algorithm	2,500 values (second)	1,000,000 values (second)
Rössler System	20.325	N/A
Lozi Map	5.9e-5	0.0132
Marsenne Twister Algorithm	9.084e-5	0.0201

8. Experimental Results: Summary

Considering only the dynamical system performance without computing time, CHAT with Lozi map can cover the highest number of nodes in the ring and random topologies. On the other hand, Rössler CHAT shows exceptional results in small world and power-law topologies. Both chaotic systems yield comparable Coverage Percentage on the grid topology. Apart from the Coverage Percentage which defines the number of discovered nodes of an algorithm, we also propose the Mean Time Coverage metric to measure the efficiency (against time) of an algorithm. When both metrics are considered in conjunction with the computation time of the Lozi map, Lozi CHAT is even better than the Rössler CHAT due to its comparable Coverage Percentage and significantly lower computation time. Furthermore, in this work, we use a bifurcation diagram of a Rössler system to further improve the performance of CHAT. The bifurcation diagram is divided into parts to capture the essence of the Rössler system including the banded chaos attractors. We also pointed out that the complexity of the banded chaos contributes to the increased performance of CHAT. The methodology we propose using this bifurcation diagram enables us to test nonequivalent chaotic mechanisms for a given optimization algorithm and other optimization problems as well. This partition method can also be used with the Lozi map to avoid the gap in the value sequence which is unwanted in our use case. However, the topological analysis cannot be performed to compare the chaotic mechanisms of the Lozi map.

Table 8. Best parameters for Lozi CHAT for each tested graph topology

Topology	Lozi map parameters	Algorithm
Ring	$a = 1.5$, $b = -0.44$	Vanilla
Small World	$a = 1.5$, $b = -0.42$	Vanilla
Random	$a = 1.78$, $b = 0.1$	Circular Last
Grid	$a = 1.43$, $b = 0.1$	Circular Last
Power-Law	$a = 1.5$, $b = 0.41$	Circular Last

According to the previous statement, Tab. 8 summarizes the best parameters and CHAT algorithms on each topology to use with the Lozi map. Therefore, we aim to further improve the current algorithm into a multi-agent system which can employ several chaotic system at once and analyze the result through templates and subtemplates [Rosalie, 2016]. With the reported performance of the Lozi map being comparable to the Rössler system and taking less time to solve the system, it is worth studying it further and if possible, extract similar continuous chaotic system behaviors out of the discrete systems. The work presented in [Aziz-Alaoui *et al.*, 2001] is very promising in this regard. Moreover, some recent tools developed to obtain chaotic flow with suspension of the map (including the template of the chaotic dynamics) can be used to explore the capabilities of the Lozi map [Starrett, 2012; Mangiarotti & Letellier, 2015]. All of these findings and prospects demonstrate the capability to shorten the computation time of the chaotic dynamics and preserve the same Coverage Percentage in the algorithm. They also open many possibilities in experimenting with various existing chaotic systems.

9. Conclusion

In this paper, we present a novel chaotic agent-based graph traversal algorithm called "Chaotic traversal (CHAT)" that focuses on a large graph traversal. In most cases, the information about large real-world graphs is not readily available, hence, we use the Coverage Percentage as the efficiency index in this work. Since this is a chaotic-based algorithm, we employs two chaotic dynamics in our CHAT algorithm: Lozi map and Rössler system. The performances of the two dynamics are also compared in this work using the Coverage Percentage and Mean Time Coverage. The algorithm is tested on five topologies against random walk, ring, small world, random, grid and power-law. Two versions of CHAT are proposed, Vanilla and Circular (First and Last) versions. The Vanilla version is totally memoryless, while the Circular versions utilize a constant and negligible amount of memory to mitigate the rare occurrence of loop traversal problem

in the Vanilla version. Both methods are compared against the Random Walk algorithm. The results show that CHAT is better than the Random Walk in both Coverage Percentage and Mean Time Coverage. Based on this conclusion, we propose a set of parameters of the Lozi map to obtain the best results from CHAT algorithms on all tested topologies. With all these results, our CHAT algorithm is more suitable in exploring the unknown parts of the social network, World Wide Web graphs and biological graphs as small world property and power-law distribution are found in them. Not to mention that the size of these graphs are big, our CHAT algorithm that requires very little memory to no memory at all is a very suitable algorithm for big unknown graphs exploration.

Regarding to the work in Chaos theory, for continuous chaotic system our current work only permits us to study genus–1 torus attractors, we aim to improve the methodology to be able to describe chaotic attractors bounded by higher genus torus such as the Lorenz attractor [Lorenz, 1963], Chen attractor [Chen & Ueta, 1999] and Chua attractor [Chua *et al.*, 1993]. This idea is indeed possible since we already know that the first return maps associated to these attractors can provide symmetric structures with additional transitions between components [Rosalie & Letellier, 2014]. Apart from being a novel graph traversal algorithm, this work also serves as a milestone to further extend the application of chaotic dynamics to other fields as well. We will then move toward a knowledge discovery in a large graph. We plan to apply CHAT to biology graphs. One of the targeted graphs is "Parkinson Disease Map" [Fujita *et al.*, 2014], which is the graph that represents interaction between substances and proteins that could cause the Parkinson disease. This graph has an ability to grow dynamically from further collaborations between institutes which makes it a good case study because it can be large and unknown at the same time. One of the main objectives is to cluster the graph to help biologists finding anomalies and discovering overlooked information. With the properties of being an unknown and large graph, CHAT is definitely a promising candidate to help with the exploration process before the clustering process takes place.

Acknowledgments

The authors would like to express their gratitude toward the High Performance Computing (HPC) Service of University of Luxembourg for providing a computing platform to carry this research [Varrette *et al.*, 2014] (see http://hpc.uni.lu). The authors also would like to thanks the reviewers for their useful comments and remarks that improve this article.

References

Adamic, L. A. [1999] "The small world web," *ECDL*, pp. 443–452.

Albers, S. & Henzinger, M. R. [2000] "Exploring unknown environments," *SIAM J. Comput.* **29**, 1164–1188.

Albert, R. [2005] "Scale-free networks in cell biology," *J. Cell Sci.* **118**, 4947–4957.

Albert, R., Jeong, H. & Barabási, A.-L. [1999] "Internet: Diameter of the world-wide web," *Nature* **401**, 130–131.

Alon, N., Bruck, J., Naor, J., Naor, M. & Roth, R. M. [1992] "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs," *IEEE Trans. Inf. Theory* **38**, 509–516.

Araujo, E. & Coelho, L. d. S. [2008] "Particle swarm approaches using Lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system," *Appl. Soft Comput.* **8**, 1354–1364.

Aziz-Alaoui, M., Robert, C. & Grebogi, C. [2001] "Dynamics of a Hénon–Lozi-type map," *Chaos, Solitons & Fractals* **12**, 2323–2341.

Barabasi, A.-L. & Oltvai, Z. N. [2004] "Network biology: understanding the cell's functional organization," *Nat. Rev. Genet.* **5**, 101.

Botella-Soler, V., Castelo, J., Oteo, J. & Ros, J. [2011] "Bifurcations in the Lozi map," *J. Phys. A: Math. Theor.* **44**, 305101.

Bucolo, M., Caponetto, R., Fortuna, L., Frasca, M. & Rizzo, A. [2002] "Does chaos work better than noise?" *IEEE Circuits Syst. Mag.* **2**, 4–19.

Chen, G. & Ueta, T. [1999] "Yet another chaotic attractor," *Int. J. Bifurcation Chaos* **09**, 1465–1466.

Choset, H. [2001] *Ann. Math. Artif. Intell.* **31**, 113–126.

Chua, L., Wu, C., Huang, A. & Zhong, G.-Q. [1993] "A universal circuit for studying and generating chaos. II. strange attractors," *IEEE Trans. Circuits Syst I: Fund. Th. Appl.* **40**, 745–761.

Coppersmith, D., Doyle, P., Raghavan, P. & Snir, M. [1993] "Random walks on weighted graphs and applications to on-line algorithms," *J. ACM* **40**, 421–453.

Dentler, K., Guéret, C. & Schlobach, S. [2009] "Semantic web reasoning by swarm intelligence," *Proc. Int. Workshop Scalable Semantic Web Knowledge Base Systems (SSWS)*, p. 1.

Dudek, G., Jenkin, M., Milios, E. & Wilkes, D. [1991] "Robotic exploration as graph construction," *IEEE Trans. Rob. Autom.* **7**, 859–865.

Erdős, P. & Rényi, A. [1960] "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci* **5**, 17–60.

Faloutsos, M., Faloutsos, P. & Faloutsos, C. [1999] "On power-law relationships of the internet topology," *ACM SIGCOMM Computer Com. Review*, pp. 251–262.

Fleischer, R. & Trippen, G. [2003] "Experimental studies of graph traversal algorithms," *Exp. Efficient Algorithms*, pp. 120–133.

Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A. & Peleg, D. [2005] "Graph exploration by a finite automaton," *Theor. Comput. Sci* **345**, 331–344.

Fujita, K. A., Ostaszewski, M., Matsuoka, Y., Ghosh, S., Glaab, E., Trefois, C., Crespo, I., Perumal, T. M., Jurkowski, W., Antony, P. M. *et al.* [2014] "Integrating pathways of parkinson's disease in a molecular interaction map," *Mol. Neurobiol.* **49**, 88–102.

Gandomi, A. H., Yang, X.-S., Talatahari, S. & Alavi, A. [2013a] "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.* **18**, 89–98.

Gandomi, A. H., Yun, G. J., Yang, X.-S. & Talatahari, S. [2013b] "Chaos-enhanced accelerated particle swarm optimization," **18**, 327–340.

Gaudino, R., Carena, A., Ferrero, V., Pozzi, A., De Feo, V., Gigante, P., Neri, F. & Poggiolini, P. [2001] "RINGO: A WDM ring optical packet network demonstrator," *Eur. Conf. Optical Com. (ECOC'01)*, pp. 620–621.

Haenggi, M., Andrews, J. G., Baccelli, F., Dousse, O. & Franceschetti, M. [2009] "Stochastic geometry and random graphs for the analysis and design of wireless networks," *IEEE J. Sel. Areas Commun.* **27**.

Hagberg, A. A., Schult, D. A. & Swart, P. J. [2008] "Exploring network structure, dynamics, and function using NetworkX," *Proc. Python in Science Conf. (SciPy2008)*, pp. 11–15.

Hamaizia, T., Lozi, R. & Hamri, N.-e. [2012] "Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor," *Appl. Math. Comput.* **219**, 188–196.

Higashikawa, Y., Katoh, N., Langerman, S. & Tanigawa, S. [2014] "Online graph exploration algorithms for cycles and trees by multiple searchers," *J. Comb. Opt.* **28**, 480–495.

Holme, P. & Kim, B. J. [2002] "Growing scale-free networks with tunable clustering," *Phys. Rev. E* **65**, 026107.

Hong, S., Oguntebi, T. & Olukotun, K. [2011] "Efficient parallel graph exploration on multi-core CPU and GPU," *Proc. IEEE Int. Conf. Parallel Architectures and Compilation Techniques*, pp. 78–88.

Horvitz, E. & Leskovec, J. [2007] "Planetary-scale views on an instant-messaging network," *MSR-TR* .

Ilcinkas, D. [2008] "Setting port numbers for fast graph exploration," *Theor. Comput. Sci* **401**, 236–242.

Kalyanasundaram., B. & Pruhs, K. R. [1994] "Constructing competitive tours from local information," *Theor. Comput. Sci.* **130**, 125–138.

Khan, A. & Elnikety, S. [2014] "Systems for big-graphs," *Proceedings of the VLDB Endowment* **7**, 1709–1710.

Koenig, S. & Smirnov, Y. [1996] "Graph learning with a nearest neighbor approach," *Proc. ACM Annual Conf. Computational learning theory*, pp. 19–28.

Koenig, S., Szymanski, B. & Liu, Y. [2001] "Efficient and inefficient ant coverage methods," *Ann. Math. Artif. Intell.* **31**, 41–76.

Koyuncu, I., Ozcerit, A. T. & Pehlivan, I. [2014] "Implementation of FPGA-based real time novel chaotic oscillator," *Nonlinear Dynamics* **77**, 49–59.

Kruskal, W. H. & Wallis, W. A. [1952] "Use of ranks in one-criterion variance analysis," *J. Am. Stat. Assoc.* **47**, 583–621.

Kwek, S. [1997] "On a simple depth-first search strategy for exploring unknown graphs," *Lect. Notes Comput. Sci.*, pp. 345–353.

Li, L., Yang, Y., Peng, H. & Wang, X. [2006] "An optimization method inspired by "chaotic" ant behavior," *Int. J. Bifurcation Chaos* **16**, 2351–2364.

Liu, B., Wang, L., Jin, Y.-H., Tang, F. & Huang, D.-X. [2005] "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals* **25**, 1261–1271.

Liu, D. & Chen, G. [2010] "Hybrid algorithm for ant colony optimization based on chaos technique," *Proc. IEEE Int. Conf. Natural Computation*, pp. 2628–2632.

Lorenz, E. N. [1963] "Deterministic nonperiodic flow," *J. Atmos. Sci.* **20**, 130–141.

Lovász, L. [2012] *Large networks and graph limits*, Vol. 60 (Providence: American Mathematical Society).

Lozi, R. [1978] "Un attracteur étrange (?) du type attracteur de Hénon," *J. Phys.* **39**, C5–9.

Lozi, R. [2012] "Emergence of randomness from chaos," *Int. J. Bifurcation Chaos* **22**, 1250021.

Mangiarotti, S. & Letellier, C. [2015] "Topological analysis for designing a suspension of the Hénon map," *Phys. Lett. A* **379**, 3069–3074.

Matsumoto, M. & Nishimura, T. [1998] "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM T. Model Comput. S.* **8**, 3–30.

Mitzenmacher, M. [2004] "A brief history of generative models for power law and lognormal distributions," *Internet Math.* **1**, 226–251.

Moravec, H. & Elfes, A. [1985] "High resolution maps from wide angle sonar," *Proc. IEEE Int. Conf. Rob. Automation*, pp. 116–121.

Murphy, R., Berry, J., McLendon, W., Hendrickson, B., Gregor, D. & Lumsdaine, A. [2006] "DFS: A simple to write yet difficult to execute benchmark," *Proc. IEEE Int. Symp. Workload Characterization*, pp. 175–177.

Newman, M. E., Strogatz, S. H. & Watts, D. J. [2001] "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E* **64**, 026118.

Panaite, P. & Pelc, A. [1999] "Exploring unknown undirected graphs," *J. Algorithms* **33**, 281–295.

Piyatumrong, A., Bouvry, P., Guinand, F. & Lavangnananda, K. [2009a] "A study of token traversal strategies on tree-based backbones for mobile ad hoc - delay tolerant networks," *Proc. IEEE Int. Conf. Ultra Modern Telecommunications (ICUMT)*, pp. 1–8.

Piyatumrong, A., Ruiz, P., Bouvry, P., Guinand, F. & Lavangnananda, K. [2009b] "Token traversal strategies of a distributed spanning forest algorithm in mobile ad hoc - delay tolerant networks," *Proc. Int. Conf. Advances in Information Technology (IAIT)* (Springer), pp. 96–109.

Pluhacek, M., Senkerik, R. & Zelinka, I. [2015] "PSO algorithm enhanced with Lozi chaotic map-tuning experiment," *AIP Conf. Proc.*, p. 550022.

Richter, A., Bock, H., Fischler, W., Leisching, P., Krummrich, P., Mayer, A., Neuhauser, R., Elbers, J. & Glingener, C. [2001] "Germany-wide DWDM field trial: transparent connection of a long haul link and a multiclient metro network," *Optical Fiber Com. Conf.*, p. ML3.

Rosalie, M. [2016] "Templates and subtemplates of Rössler attractors from a bifurcation diagram," *J. Phys. A: Math. Theor.* **49**, 315101.

Rosalie, M., Danoy, G., Chaumette, S. & Bouvry, P. [2016] "From random process to chaotic behavior in swarms of UAVs," *Proc. ACM Int. Symp. on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet 2016)*, pp. 9–15.

Rosalie, M. & Letellier, C. [2014] "Toward a general procedure for extracting templates from chaotic attractors bounded by high genus torus," *Int. J. Bifurcation Chaos* **24**, 1450045.

Rössler, O. E. [1976] "An equation for continuous chaos," *Phys. Lett. A* **57**, 397–398.

Sanz-Arigita, E. J., Schoonheim, M. M., Damoiseaux, J. S., Rombouts, S. A., Maris, E., Barkhof, F., Scheltens, P. & Stam, C. J. [2010] "Loss of 'small-world' networks in alzheimer's disease: graph analysis of fmri resting-state functional connectivity," *PloS one* **5**, e13788.

Seaton, K. A. & Hackett, L. M. [2004] "Stations, trains and small-world networks," *Physica A* **339**, 635–644.

Servetto, S. D. & Barrenechea, G. [2002] "Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks," *Proc. ACM Int. Workshop Wireless sensor networks and applications*, pp. 12–21.

Shrikhande, K. V., White, I. M., Wonglumsom, D.-R., Gemelos, S. M., Rogge, M. S., Fukashiro, Y., Avenarius, M. & Kazovsky, L. G. [2000] "HORNET: A packet-over-WDM multiple access metropolitan area ring network," *IEEE J. Sel. Areas Commun.* **18**, 2004–2016.

Sprott, J. & Li, C. [2017] "Asymmetric bistability in the Rössler system," *Acta Phys. Pol. B* **48**, 97.

Starrett, J. [2012] "Non-strange chaotic attractors equivalent to their templates," *Dyn. Sys.* **27**, 187–196.

Tiddi, I., d'Aquin, M. & Motta, E. [2014] "Walking linked data: a graph traversal approach to explain clusters," *Proc. Int. Conf. Consuming Linked Data-Volume 1264*, pp. 73–84.

Varrette, S., Bouvry, P., Cartiaux, H. & Georgatos, F. [2014] "Management of an academic HPC cluster: The UL experience," *Proc. Int. Conf. High Performance Computing & Simulation (HPCS 2014)*, pp. 959–967.

Wagner, A. & Fell, D. A. [2001] "The small world inside large metabolic networks," *Proc. R. Soc. London, Ser. B* **268**, 1803–1810.

Wagner, I. A., Lindenbaum, M. & Bruckstein, A. M. [1999] "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Rob. Autom.* **15**, 918–933.

Watts, D. J. & Strogatz, S. H. [1998] "Collective dynamics of small-world networks," *Nature* **393**, 440–442.

Wilcoxon, F. [1945] "Individual comparisons by ranking methods," *Biometrics bulletin* **1**, 80–83.

Wong, A. K. & You, M. [1985] "Entropy and distance of random graphs with application to structural pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* , 599–609.

Xiang, T., Liao, X. & Wong, K.-W. [2007] "An improved particle swarm optimization algorithm combined with piecewise linear chaotic map," *Appl. Math. Comput.* **190**, 1637–1645.

Yanovski, V., Wagner, I. A. & Bruckstein, A. M. [2001] "Vertex-ant-walk–a robust method for efficient exploration of faulty graphs," *Ann. Math. Artif. Intell.* **31**, 99–112.

Yook, S.-H., Oltvai, Z. N. & Barabási, A.-L. [2004] "Functional and topological characterization of protein interaction networks," *Proteomics* **4**, 928–942.