UTRECHT UNIVERSITY

MASTER THESIS

FACULTY OF SCIENCE

Generating Maximal Independent Sets Using Lotka-Volterra Dynamics

Author
M. N. MOOIJ

Supervisor
Dr. I. KRYVEN

Second Readers
Dr. P. SALANEVICH
Dr. C. GROENLAND

Utrecht University

June 2022

Abstract

Systems of ordinary differential equations (ODEs) are rarely thought of as a means of discrete computations. We consider finding the maximum independent set in a graph which is known to be a computationally demanding (NP-hard) problem. We show that one can construct an approximate solution to this problem by exploring the stable manifold of a particular system of ODEs by using the numerical continuation. Interestingly, our system of ODEs can be regarded as the Lotka-Volterra dynamics for competing biological species with a binary interaction matrix, which suggests a parallel with natural computing.

Contents

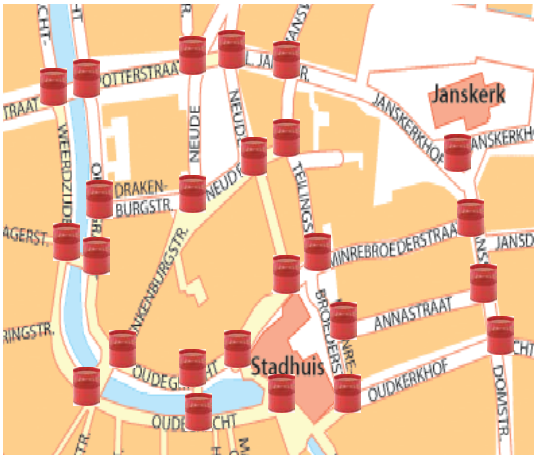
Introduction

In 1763, Leonhard Euler published what is widely regarded as the first paper in graph theory. This paper is about the well-known Seven Bridges of Königsberg Problem [46]. Since then, a lot of work has been done in this area. Applications of graph theory have been found in service industry scheduling [1], school bus routing [2], computer communication [2], wireless network planning [12, 40], error correcting codes [12], image and data processing [44], and many other areas.

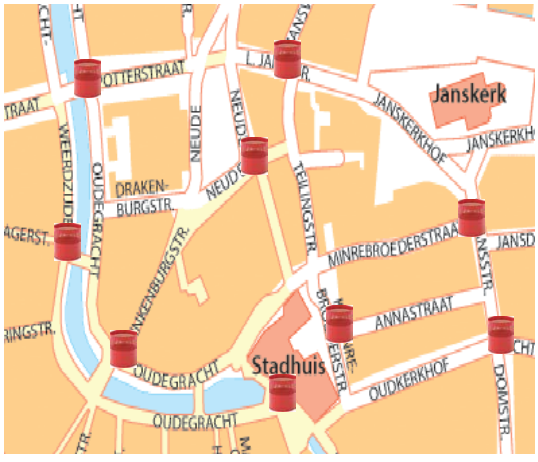
Eventually research gave rise to the notion of NP-hard problems. These are problems which can not be solved in a "reasonable" amount of time. A list with many of these problems can be found in [17].

A well-known NP-hard problem is the *Maximum Independent Set (MIS) Problem* [17]. This problem has many applications, for example in radio network optimization [13]. This problem aims to find for $G = (V, E)$ a simple graph the largest subset $A \subseteq V$, such that for every $v \in A$, we have for $w \in N(v)$ that $w \notin A$, where $N(v)$ denotes the neighbors of v.

Application
Assume the city of Utrecht wants to place new garbage bins in the city center. They want to maximize the number of placed garbage bins. Preferably, they want to place a garbage bin at every street crossing. The company that empties the garbage bins thinks this is a bad idea, since this would provide them with more work and they do not have the appropriate resources available. They pose the following constraint; If a garbage bin is placed at a street crossing, there can not be a garbage bin at an adjacent street crossing. The cite poses the constraint that they want to maximize the number of garbage bins that are placed.



(a) Too many garbage bins.



(b) Possible configuration.

Finding the configuration such that the most garbage bins are placed means finding a maximum independent set in the street network of Utrecht.

There are many algorithms that solve the MIS problem exactly. For example, one can look at the $O(2^{0.276n})$-time algorithm introduced by J. Robson [38] or the $1.1996^n n^{O(1)}$-time polynomial-space algorithm of M. Xiao and H. Nagamochi [47].

Since the MIS problem is NP-hard, it is not (yet?) possible to solve it exactly in polynomial time. It is possible however, to solve it approximately in polynomial time [4, 5, 6, 7, 10, 23, 24, 36]. The aim of this thesis is to find new approximation algorithms, using Lotka-Volterra (LV) dynamics, given by the LV equations [34].

We propose two algorithms that generate maximal independent sets in a graph. Both of these algorithms are based on the LV equations, taken from ecology. These equations are a set of ordinary differential equations, used to model behavior of species in ecological systems.

- In the first algorithm we investigate the trajectories of the LV equations

$$\frac{d\mathbf{x}}{dt} = diag(x)(\mathbf{r} + A\mathbf{x}), \tag{1}$$

  with parameters $\mathbf{r} = \mathbf{1}$ and $A = \tau A^0 + \mathbb{I}$, for $\tau > 1$ and A^0 the adjacency matrix of a graph G. We see that these trajectories can be used to determine a maximal independent set in the graph G.

- In the second algorithm we again use the Lotka-Volterra equations. Instead of looking at the trajectories of the corresponding dynamical system, we use numerical continuation to investigate the behavior of the fixed points under the change of parameters. It turns out that with this approach we are able to find a maximal independent set in a graph.

The novelty of this approach is that we have biological heuristic to explain the behavior of both algorithms. This allows us to solve a discrete problem by means of a continuous system. We discuss the link between ecology and the MIS problem in Section 5. This link allows us to use statements from ecology to describe the behavior of the algorithm.

An approach that also uses a continuous system to solve a discrete optimization problem is given by Bomze et al [9]. This approach uses the replicator equations, another ecological system, to approximate the Maximum Clique problem. We discuss this approach in Section 5.

Reading Guide

A goal of this thesis is to be self-contained. To reach this goal, we use four sections to introduce material that is necessary to discuss the proposed algorithms. One can also see these first four sections as an introduction to the thesis. If one is familiar with this material, it can be skipped.

- In Section 1 of this thesis, we give an introduction into graph theory. We discuss basic notions along with general information about graphs. We use these notions later in the thesis.
- In Section 2, we discuss random graphs and how to construct them. We observe that the eigenvalue spectrum of random regular graphs is given by the McKay distribution.
- In Section 3, we give an introduction into complexity theory and NP-complete problems. We show the MIS problem is NP-complete and we give examples of an exact and an approximation algorithm.
- In Section 4, we discuss the LV dynamics and the ecological implications.
- In Section 5, we show that for appropriate parameters the LV dynamics converge to a maximal independent set.
- In Section 6, we introduce a new algorithm inspired by ecological heuristics of the LV dynamics and numerical continuation.
- In Section 7, we do simulations to determine the performance of the two algorithms. We will see that, for certain graph types, the second algorithm (Continuation) performs better than the first algorithm (LV).

A star ($*$) will be used to mark new results first obtained in this thesis (to the best of the author's knowledge), in order to distinguish them from already existing results in the literature.

1 Graph Theory

Graph theory is a very active discipline in mathematics. It has applications in service industry scheduling [1], school bus routing [2], computer communication [2], error correcting codes [12], wireless network planning [12, 40], image and data processing [44], and many other areas. In this section, we first show basic but necessary graph theoretical notions. For a more detailed discussion of the basics of graph theory, the reader is referred to [22]. Then we discuss some important graph families. Finally, we look at spectral properties of graphs.

1.1 Basic notions

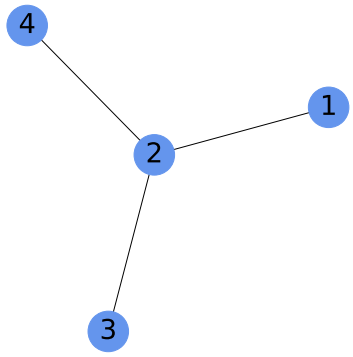
In this thesis, we denote $[n] := \{1, 2, \dots, n\}$ to simplify the notation.

Definitions:

- Graph: A *graph* $G = (V, E)$ consists of a set vertices $V = \{v_i \ : \ i \in [n]\}$ and a set edges $E \subseteq V \times V$. With the size of the graph $|G|$, we denote the number of vertices $|V|$ in the graph.
- Adjacency matrix: We define the *adjacency matrix* of a graph as

$$A^0_{ij} = \begin{cases} 1 & \text{if there is an edge between vertex } v_i \text{ and vertex } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

- (Un)directed: We call a graph *undirected* if $A^0_{ij} = A^0_{ji}$ for every $i, j \in [n]$. If not, we call the graph *directed*.
- Neighborhood: Let $G = (V, E)$ be a graph. For any vertex $v_i \in V$, we denote the *neighborhood* of v_i by $N(v_i) := \{v_j \in V \backslash \{v_i\} \ : \ A^0_{ij} = 1\}$.
- Degree: The *degree* of a vertex $v_i \in V$ is the number of neighbors of v_i. We denote the degree by $d_i = |N(v_i)|$.



$$A^0 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

vertex	degree
v_1	1
v_2	3
v_3	1
v_4	1

Figure 1.1: Graph example.

- Walk: A *walk* on a graph is a sequence of vertices, such that every pair of consecutive vertices is connected by an edge.
- Path: A *path* in a graph is a walk where every vertex occurs only once.
- Cycle: A walk in which the first vertex is the same as the last vertex and every other vertex occurs at most once, we call a *cycle*.
- Connected: A graph is *connected* if there exists a path between any two distinct vertices in the graph.
- Diameter: Let $G = (V, E)$ be a connected graph. The *diameter* m of G is defined as the longest shortest path between vertices of G. That is, $m = \max\{|P(v_i, v_j)| \ : \ v_i, v_j \in V\}$, where $|P(v_i, v_j)|$ denotes the length of the shortest path between vertex v_i and v_j.

Lemma 1.1.1 (Number of walks)**.** *Let $v_i, v_j \in V$ be two vertices. The number of walks of length k from vertex v_i to vertex v_j is given by the entry $(A^0)^k_{ij}$.*

Proof. By definition of the adjacency matrix, the statement is true for $k = 1$. Assume the statement is true for some $k \in \mathbb{N}_{>1}$. Then $(A^0)^k_{il}$ equals the number of walks from vertex v_i to vertex v_l. Note $(A^0)^k_{il} A^0_{lj}$ is nonzero only when vertex v_l is a neighbor of vertex v_j. Hence,

$$\begin{aligned}(A^0)^{k+1}_{ij} &= ((A^0)^k A^0)_{ij} \\ &= \sum_{l \in [n]} (A^0)^k_{il} A^0_{lj}\end{aligned}$$

is exactly the number of walks of length $k+1$ from vertex v_i to vertex v_j. □

1.2 Different graph types

We call a graph *simple* if it is undirected and contains no self-loops or parallel edges. In this thesis, we only consider simple graphs. Therefore, when we talk about a graph, we implicitly mean a simple graph.

There are a lot of different simple graphs. We give some of the most well-known examples.

Bipartite graph: We call a graph bipartite if we can make a partition of the vertex set such that there are no edges between vertices of the same set.

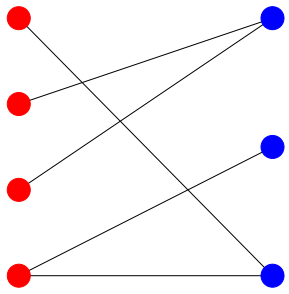


Figure 1.2: Bipartite graph.

Tree graph: We call a graph a tree if every pair of vertices is connected by exactly one path.

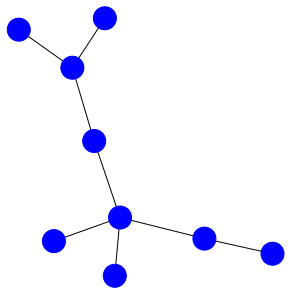


Figure 1.3: Tree graph.

Star graph: We call a graph G a star if it is a tree with one vertex having degree $|G| - 1$ and the other vertices having degree 1.

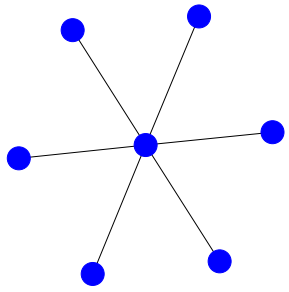


Figure 1.4: Star graph.

Regular graph: We call a graph regular if the degrees of all vertices are equal.

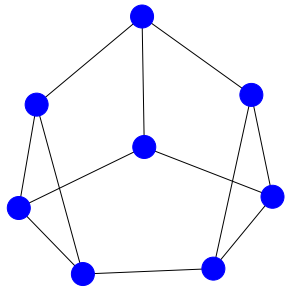


Figure 1.5: Regular graph.

Complete graph: We call a graph complete if there is an edge between any two distinct vertices of the graph.

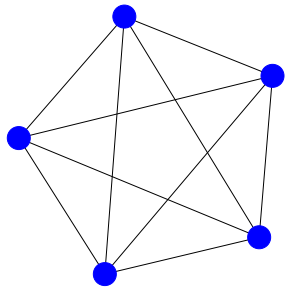


Figure 1.6: Complete graph.

Complement graph: Let $G = (V, E)$ be a graph. With the complement graph $\overline{G}$ we denote the graph consisting of vertex set V and edge set $\overline{E}$.

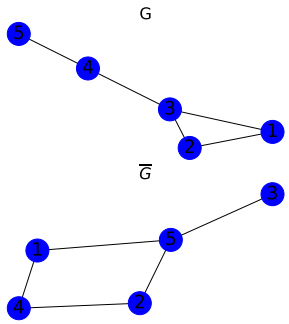


Figure 1.7: Graph G and its complement $\overline{G}$.

1.3 Spectrum of graphs

By the spectrum of a graph we mean the spectrum of the corresponding adjacency matrix. The spectrum of a graph is important in many applications. For example in finding communities [43, p.90–91] and finding minimum cuts in a graph [35]. Also, as we will see in Section 5, the spectrum of a matrix is important in determining the stability of fixed points in dynamical systems. If the spectrum of a matrix is only real-valued, we number the eigenvalues in decreasing order, that is $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Note that these eigenvalues are not necessarily unique. In the rest of this section, we consider a graph with n vertices and m edges. We denote by $d_{\max}$ and $d_{\min}$ the biggest and smallest degree of the graph.

Definition 1.3.1 (Hermitian matrix)**.** We call a matrix A *hermitian* if $A^* := \overline{A^T} = A$, where A^T is the transpose of matrix A and $\overline{A}$ is the component-wise complex conjugate of matrix A.

Lemma 1.3.2. *For every hermitian matrix, the eigenvalue spectrum is real.*

Proof. Let A be a hermitian matrix. Let λ be an eigenvalue of A with corresponding eigenvector v. Then by definition, we have $Av = \lambda v$. Hence, we get the equality

$$\lambda\|x\| = \lambda x^* x = x^* \lambda x = x^* A x = (x^* A^* x)^* = (x^* A x)^* = (\lambda\|x\|)^* = \overline{\lambda}\|x\|.$$

So we know $\overline{\lambda} = \lambda$, and therefore λ is real. □

Note that for a simple graph the adjacency matrix is symmetric and consists only of real elements. Hence, the adjacency matrix is hermitian. We can conclude by Lemma 1.3.2 that the spectrum of a simple graph is real.

Lemma 1.3.3. *Let $G = (V, E)$ be a simple graph with $|E| \neq 0$. Then the spectrum of G contains positive and negative eigenvalues.*

Proof. One way to see this is by looking at the trace of A^0. This trace is equal to the sum of eigenvalues and for a simple graph this equals zero. Therefore, either all eigenvalues are zero or there exist both positive as well as negative eigenvalues. Since $|E| \neq 0$, it follows that $A^0 \neq 0$ and, thus, it must have a nonzero eigenvalue. □

1.4 Gershgorin Circle Theorem

A possible way to bound the eigenvalues of a matrix is given by the Gershgorin circle theorem. It was first introduced in 1931 by Soviet mathematician S. Gershgorin [18]. To formulate the theorem, we introduce the Gershgorin disc.

Definition 1.4.1 (Gershgorin disc)**.** Let A be a $n \times n$ complex matrix. We define $r_i(A) = \sum_{i \neq j} |A_{ij}|$ as the sum of the magnitudes of the non-diagonal entries of the ith row. The ith *Gershgorin disc* is the disc $D(A_{ii}, r_i(A))$, centered at A_{ii} on the complex plane, with radius $r_i(A)$.

Theorem 1.4.2 (Gershgorin Circle Theorem [18])**.** *Every eigenvalue of a matrix lies within at least one of its Gershgorin discs.*

Proof. Let v be an eigenvector of A with corresponding eigenvalue λ. Since $v \neq 0$, we know $\max_{i \in [n]} |v_i| = |v_k| > 0$ for some $k \in [n]$. The kth equation of $Av = \lambda v$ gives

$$\sum_{j \neq i} A_{kj} v_j = \lambda v_k - A_{kk} v_k.$$

Therefore, we have

$$\begin{aligned} |\lambda - A_{kk}||v_k| &= |\lambda v_k - A_{kk} v_k| \\ &= |\sum_{j \neq i} A_{kj} v_j| \\ &\leq \sum_{j \neq i} |A_{kj}||v_j| \\ &\leq r_i(A)|v_k|. \end{aligned}$$

Since $v_k \neq 0$, we know

$$|\lambda - A_{kk}| \leq r_k(A).$$

So λ belongs to the kth Gershgorin disc. □

$$A = \begin{pmatrix} -5/2 & -1/2 & 3/2 \\ 3/2 & -1/2 & 0 \\ 1 & 3/2 & 5/2 \end{pmatrix}$$

$$D_1 = D(-5/2, 2)$$
$$D_2 = D(-1/2, 3/2)$$
$$D_3 = D(5/2, 5/2)$$

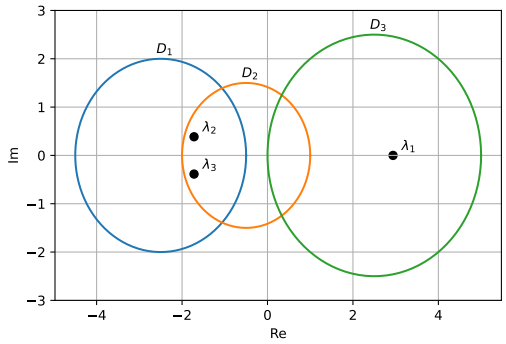


Figure 1.8: Gershgorin circle theorem example.

In Figure 1.8 we show the Gershgorin discs of an example matrix A. We also indicate the eigenvalues in the complex domain. Note all the eigenvalues are within the union of the Gershgorin discs. This is exactly what the Gershgorin circle theorem tells us. Also note that λ_1 and λ_3 are complex conjugates, something we can expect as well, due to the fact that the matrix is real-valued.

Corollary 1.4.3. *For every graph, the upper bound $\lambda_1 \leq d_{max}$ holds.*

Proof. Let A^0 be the adjacency matrix of a graph. Note that $A^0_{ii} = 0$ for all $i \in [n]$. Let v_k be a vertex with maximum degree. Note $r_i(A^0) \leq r_k(A^0)$ for all $i \in [n]$. Therefore, by Gershgorin circle theorem, we know $\lambda_i \leq d_{\max}$ for all $i \in [n]$, and in particular, $\lambda_1 \leq d_{\max}$. □

1.5 Rayleigh quotient

A characterization of λ_1 and λ_n is given by the Rayleigh quotient. Using the Rayleigh quotient, we prove a lower bound on the largest eigenvalue λ_1.

Definition 1.5.1 (Rayleigh quotient)**.** Let $A = A^*$ be a hermitian matrix. The *Rayleigh quotient* is defined by the function

$$R(x) = \frac{\langle x, Ax \rangle}{\|x\|^2}, \text{ for } x \neq 0. \tag{2}$$

Theorem 1.5.2. *[26, p.203] If A is a self-adjoint matrix, then*

$$\max_{x \neq 0} R(x) = \lambda_1, \text{ and}$$
$$\min_{x \neq 0} R(x) = \lambda_n.$$

Lemma 1.5.3. *If the vertices of G have degrees $d_1, d_2, \ldots, d_n$, then $\lambda_1 \geq \overline{d} := \frac{1}{n} \sum_{i \in [n]} d_i$.*

Proof. Using Theorem 1.5.2 with x the vector $\mathbf{1}$ with one at every coordinate, we get

$$\lambda_1 = \max_{x \neq 0} R(x) \geq R(\mathbf{1}) = \frac{1}{n} \sum_{i=1}^{n} d_i = \bar{d}.$$

□

Lemma 1.5.4. *[30, p.176] If d_{max} is the maximal degree of a vertex in G, then $\lambda_1 \geq \sqrt{d_{max}}$.*

Combining Lemma 1.4.3, Lemma 1.5.3 and Lemma 1.5.4, we get the following bound.

Proposition 1.5.5. *For every graph G,*

$$\max\{\bar{d}, \sqrt{d_{max}}\} \leq \lambda_1 \leq d_{max}. \tag{3}$$

1.6 Regular graphs

In the case of a regular graph, we can make more strict bounds on the eigenvalue spectrum. The largest eigenvalue of a regular graph can be determined as follows.

Lemma 1.6.1 (Largest eigenvalue of regular graph)**.** *For a regular graph with adjacency matrix A^0 and constant degree d, the largest eigenvalue is given by d and corresponds with the vector $\mathbf{1}$ consisting of one at every coordinate.*

Proof. Let $\mathbf{1}$ be the only ones vector. We can calculate the product with the adjacency matrix

$$\begin{aligned}(A^0\mathbf{1})_i &= \sum_{j \in [n]} A^0_{ij}\mathbf{1}_i \\ &= d.\end{aligned}$$

So we know $\lambda_1 \geq d$. Note every Gershgorin disc of the matrix A^0 is given by $D(0, d)$. So by the Gershgorin circle Theorem 1.4.2, we know $\lambda_1 \leq d$. This completes the proof. □

1.7 Bipartite graphs

We make an observation about cycles in a bipartite graph.

Lemma 1.7.1. *A graph is bipartite if and only if it has no cycles of odd length.*

Proof. Assume G is a bipartite graph. Let $A \cup B$ be the bipartite partition of the graph. Note at every step of a cycle, we have to jump from A to B or vice versa. So we can never return to the starting point of the cycle in an odd number of steps.

Assume G is a graph without cycles of odd length. We show how to partition G into two sets A and B such that there are no edges between vertices in the same set.

Let C be a connected component of G. With $d(v, w)$ we denote the distance between vertex v and vertex w. That is, the length of the shortest path between v and w. Let v be a vertex in C. For all $w \in C$, we put w in A if $d(v, w)$ is even and we put w in B if $d(v, w)$ is odd. Continue this procedure for all connected components of G.

Assume there exists an edge between $u \in A$ and $w \in A$. Then, by construction, there exists a cycle of odd length between v and w. This cycle is given by the shortest path between v and w attached to the shortest path between v and u attached to the edge between u and w. By assumption, this can not happen. Hence, the graph G is bipartite. □

2 Random Graphs

In Section 1, we discussed graph theory and bounds on the spectrum of different graphs. In this section, we look at graphs that are generated in a random way. We look at different ways to generate a random graph, which gives us different random graph types. We discuss critical bounds for connectedness of these graphs. At the end of this section, we talk about the spectrum of random regular graphs.

2.1 Different random graph models

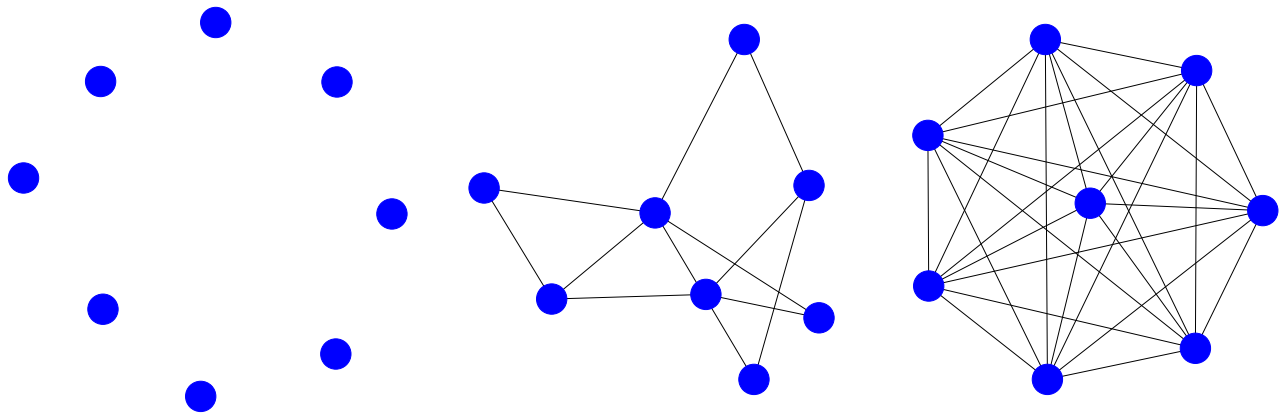
There are many types of random graphs. We show some of the most well-known examples here.

2.1.1 Erdős-Rényi graphs

The Erdős-Rényi random graphs were introduced by P. Erdős and A. Rényi in 1960 [16].

Erdős-Rényi graph
Let $n \in \mathbb{N}$ and $p \in [0, 1]$. An Erdős-Rényi graph is constructed in the following way:

1. Initialize n vertices $(v_i)_{i\in[n]}$.
2. Add each edge independently with probability p.
3. Return the obtained graph.



(a) $p = 0$. (b) $p = 0.5$. (c) $p = 1$.

Figure 2.1: Erdős-Rényi graph realizations constructed on 8 vertices for different connection probabilities.

Note that in a graph with n vertices, we can have at most $\binom{n}{2} = \frac{n(n-1)}{2}$ edges. Since we add every edge with a fixed probability p and independently of the other edges, the probability of picking a particular graph with n vertices and m edges from all possible graphs with n vertices is given by

$$p^m(1-p)^{\frac{n(n-1)}{2}-m}.$$

As shown in [16, p.57], there exists a threshold for p for which the Erdős-Rényi graph switches from unconnected to connected with high probability.

Lemma 2.1.1 (Erdős-Rényi threshold)**.** *Let $\varepsilon > 0$. If $p < \frac{(1-\varepsilon)\log n}{n}$, then a $G(n, p)$ graph is almost surely disconnected. If $p > \frac{(1+\varepsilon)\log n}{n}$, then a $G(n, p)$ graph is almost surely connected.*

In Figure 2.2, we have generated Erdős-Rényi graphs with different connection probabilities p. One can see the connectivity indeed changes when we increase the connection probability.

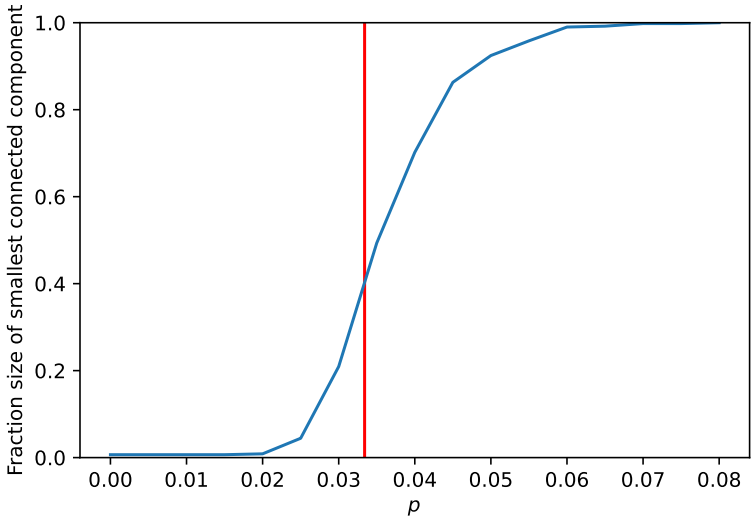


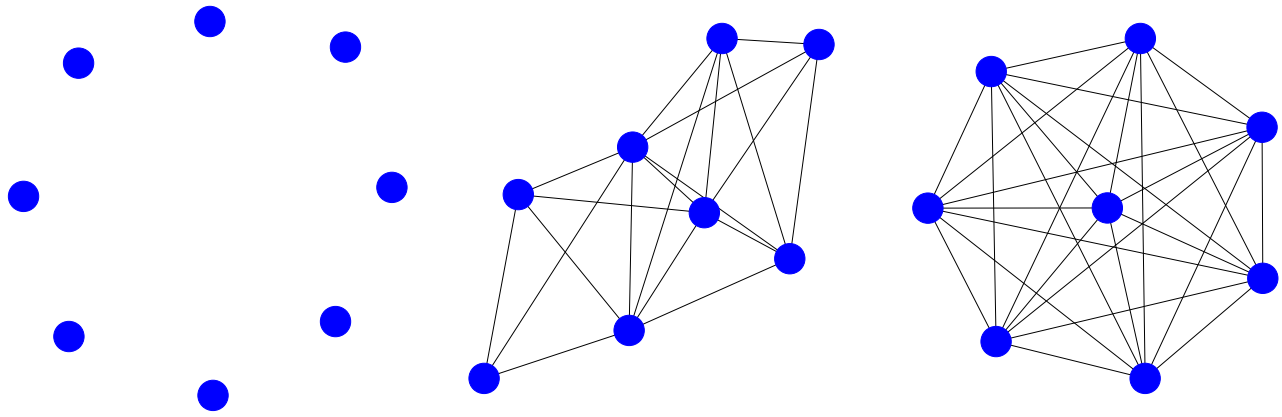
Figure 2.2: Size of the smallest connected component of a Erdős-Rényi graph of size $n = 150$. The red line indicates the critical threshold of $\log(n)/n$. Averaged over 500 runs.

2.1.2 Random geometric graphs

We can generate a random geometric graph of size n with connection parameter r in the following way.

Random geometric graph
Let $n \in \mathbb{N}$ and $r \in [0, 1]$. A random geometric graph is constructed in the following way:

1. Initialize n vertices $(v_i)_{i\in[n]}$.
2. Pick n points $(x_i)_{i\in[n]}$ uniformly at random from $(0, 1)^2$.
3. Put an edge between vertex v_i and vertex v_j if $d(x_i, x_j) < r$.
4. Return the obtained graph.



(a) $r = 0$. (b) $r = 0.5$. (c) $r = 1$.

Figure 2.3: Geometric graphs constructed on 8 vertices for different connection parameter values.

Just as for Erdős-Rényi graphs, there exists a critical threshold for the parameter r, for which the connectivity of the geometric graph changes with high probability.

For $r < \sqrt{\frac{\log n}{\pi n}}$, a geometric graph of size n with connection parameter r is almost surely not connected [32, p.120]. In Figure 2.4 we have generated geometric random graphs with different connection parameters r. One can see the connectivity indeed changes when we increase the connection probability.

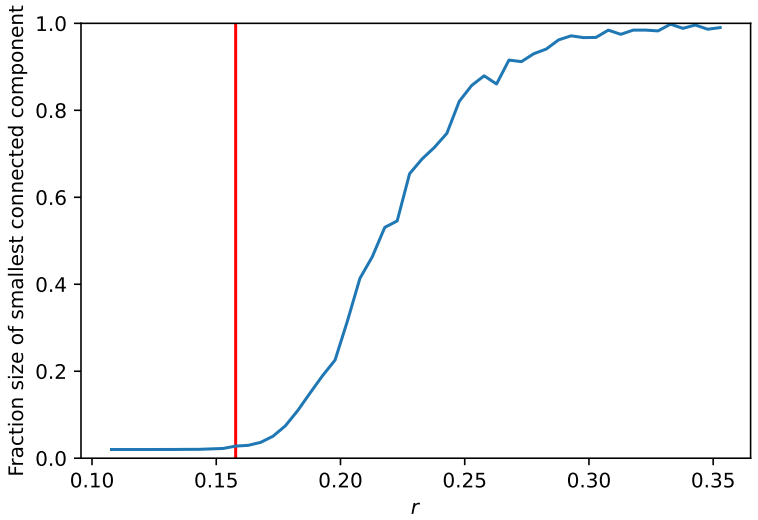


Figure 2.4: Size of the smallest connected component of a geometric(r,n) graph of size $n = 50$. The red line indicates the critical threshold of $\sqrt{\frac{\log n}{\pi n}}$. Averaged over 500 runs.

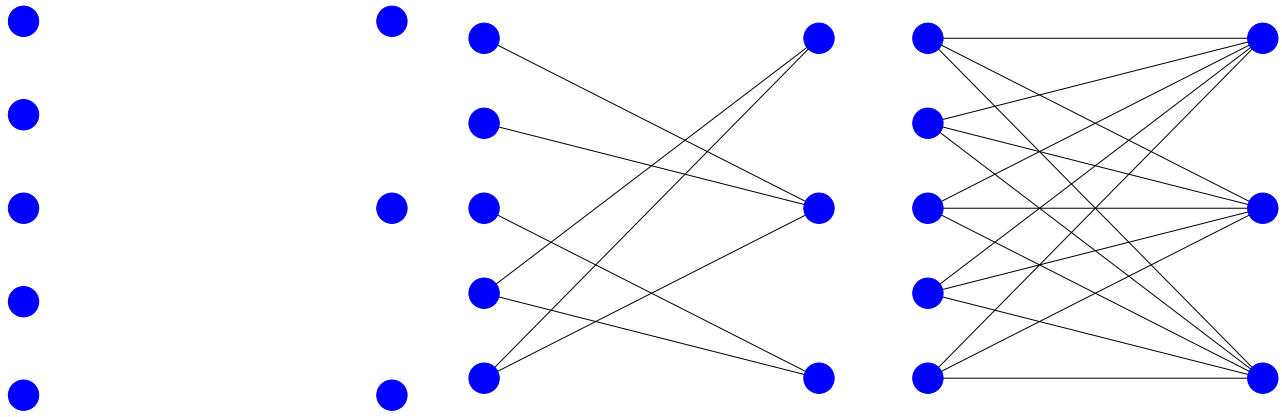
2.1.3 Random bipartite graphs

In order to generate random bipartite graphs, we can do a similar procedure as for a Erdős-Rényi graph. Note that a bipartite graph consists of two sets A and B.

We can generate a random bipartite graph with components A and B by adding every possible edge from set A to set B with a probability p.

Random bipartite graph
Let $n \in \mathbb{N}$ and $p \in [0, 1]$. A random bipartite graph is constructed in the following way:

1. Initialize n vertices $(v_i)_{i \in [n]}$.
2. Split up the vertices in the required bipartite sizes $|A|$ and $|B|$.
3. Add each edge between A and B independently with probability p.
4. Return the obtained graph.



(a) $p = 0$. (b) $p = 0.5$. (c) $p = 1$.

Figure 2.5: Random bipartite graphs constructed on 8 vertices for different connection probabilities.

2.2 McKay distribution

Random regular graphs can be generated uniformly at random in time-complexity $O(nd^2)$, where n is the number of vertices in the graph and d the constant degree of the graph [29]. In the case of random regular graphs, a lot is known about the distribution of eigenvalues of the adjacency matrix. In 1981, the spectral density of the adjacency matrix of random regular graphs was determined by B. McKay [33].

Theorem 2.2.1 (McKay distribution)**.** *Let $X_1, X_2, \ldots$ be a sequence of regular graphs with degree $d \geq 2$ such that $|X_i| \to \infty$ and $c_k(X_i)/|X_i| \to 0$ as $i \to \infty$ for each $k \geq 3$, where $c_k(X_i)$ is the number of k-cycles in X_i. The spectral density for the eigenvalues of X_i as $i \to \infty$ is given by*

$$f(x) = \begin{cases} \frac{d\sqrt{4(d-1)-x^2}}{2\pi(d^2-x^2)} & \text{for } |x| \leq 2\sqrt{d-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

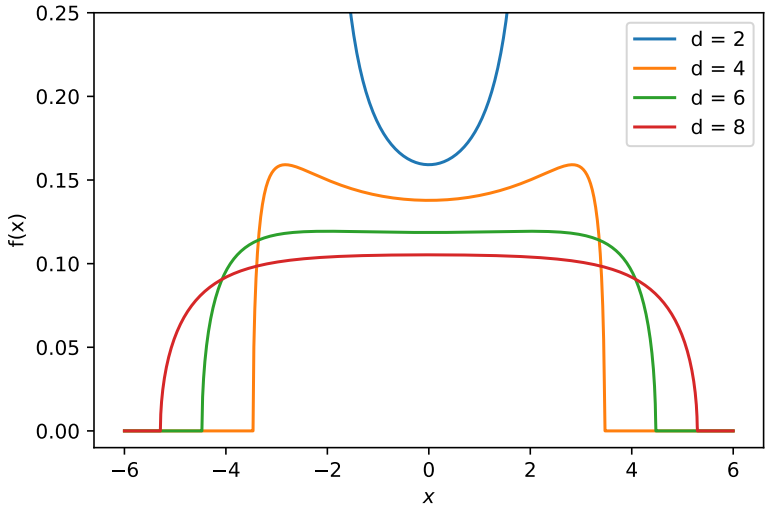


Figure 2.6: McKay distribution for regular graphs with different degrees.

In [33, p.214], it is shown that random regular graphs obey the requirement $c_k(X_i)/|X_i| \to 0$ as $i \to \infty$, for each $k \geq 3$. Therefore, we can use Theorem 2.2.1 to determine the spectral density of random regular graphs.

In Figure 2.6, one can see the McKay probability distribution for random regular graphs with different degrees. Note that this does not imply that an adjacency matrix can not have an eigenvalue bigger than $2\sqrt{d-1}$. What it does say is that the limiting portion of the eigenvalues larger than $2\sqrt{d-1}$ is zero. In fact, we have already seen one eigenvalue that is larger than $2\sqrt{d-1}$. Since we are looking at regular graphs, we know d is always an eigenvalue.

The McKay distribution gives us a way to make statements about almost all (in a probabilistic sense) eigenvalues of random regular graphs. We use this distribution in Section 5.5.

3 Complexity Theory

We use this section to introduce optimization problems and different kinds of algorithms to solve these. The work we do in this section is needed to understand the MIS problem and the algorithms we introduce later in this thesis.

3.1 Complexity theory

Arguably, one of the most wide-known mathematical problems in and outside mathematics is the problem P vs. NP. This problem is subject of much mathematical research and discussion [14, 45]. You can even earn a million dollars if you solve it. But what makes this problem so difficult? First we try to understand the problem by providing an example.

What if someone gives us a wallet filled with money. This person wants to know how much money is in the wallet. In this case, a suitable algorithm to solve this problem is to simply count the money one by one. Let's say the wallet contains five dollar bills and we can count one dollar bill in one second. Then this algorithm takes 5 seconds. If the wallet contains 20 bills, the algorithm takes 20 seconds. This is an example of a polynomial (P) time algorithm. The computation time is a polynomial function of the size n of the wallet (n^1 in this case).

Now what if someone gives us a 4-digit lock. Suppose we get a 4-digit code and the question is to check if the code opens the lock. We can easily try this on the lock. Even if we have a 1000-digit lock, this problem can be solved in polynomial time (again order n^1 here).

But what if the question is how to open the lock. In the case of the 4-digit lock, we would need to check 10^4 possibilities. If checking a possibility takes 1 second, this takes us 10^4 seconds, that is 2.8 hours. If this still seems manageable, think about when the lock has a 1000-digit code. Cracking this code takes 10^{1000} seconds, that is $3.17 \cdot 10^{989}$ millennia. Perhaps not possible anymore, assuming you want to be home before dinner. This type of problem we call a NP-hard problem.

The problem P=NP is the question if these problems are actually the same. Are NP-problems really impossible to solve in polynomial time or is it just that we have not found the correct polynomial time algorithm yet.

In this section, we first introduce different optimization problems on graphs. Then we give a rigorous definition of decision problems and how we can relate these to the optimization problems we have discussed. We prove that these optimization problems are actually NP-hard problems.

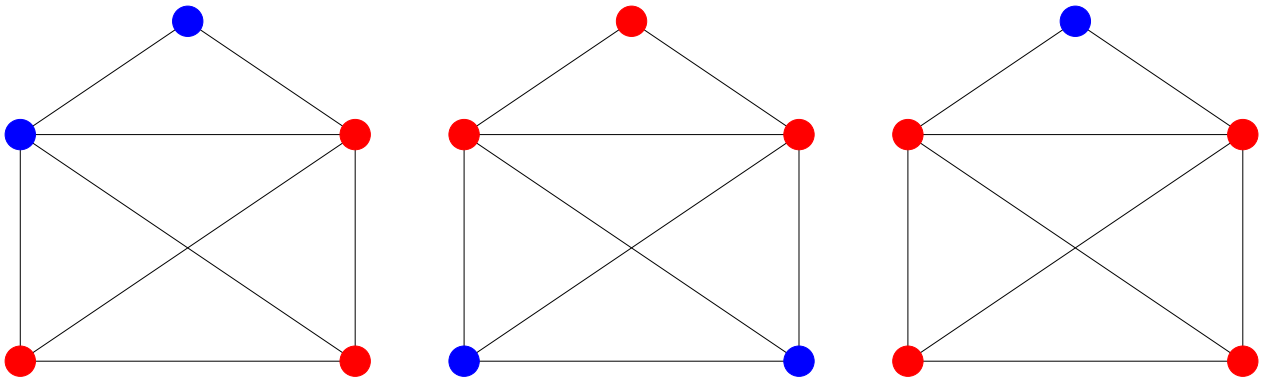
3.2 Optimization problems on graphs

Here, we describe some optimization problems on graphs. We use the same definition of graphs we have encountered in Section 1. The reason we introduce several of these optimization problems is because we need them in our reasoning in later sections.

3.2.1 Maximum Clique problem

Definition 3.2.1 (Clique)**.** Let G be a graph with vertex set V and edge set E. A *clique* is a set $C \subseteq V$ such that for each $v, w \in C$ there exists an edge that connects v and w.

Definition 3.2.2 (Maximal/Maximum clique)**.** Let G be a graph with vertex set V and edge set E. Let C be a clique of G. We call the clique C *maximal* if there exists no other clique C' of G such that $C \subset C'$. We call a maximal clique of G that has the largest cardinality a *maximum clique* of G.



(a) Clique. (b) Maximal clique. (c) Maximum clique.

Figure 3.1: Cliques in a graph.

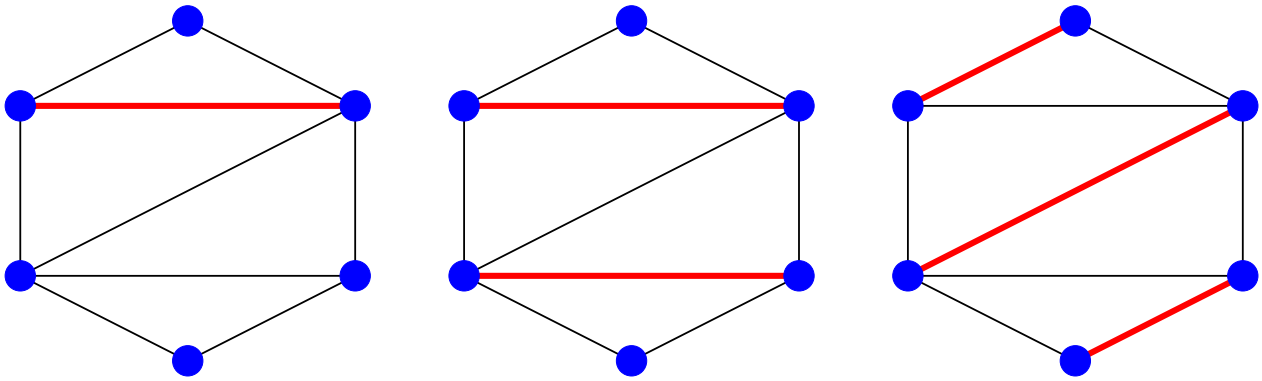
An example of a clique in a graph is given by social networks formed by friendships. In this case, a maximum clique in the graph is given by a biggest group of friends.

Solving the *Maximum Clique* problem means finding a maximum clique in a graph.

3.2.2 Maximum Matching problem

Definition 3.2.3 (Matching)**.** Let G be a graph with vertex set V and edge set E. A *matching* is a set $M \subseteq E$ such that no edges in M share the same endpoint.

Definition 3.2.4 (Maximal/Maximum matching)**.** Let G be a graph with vertex set V and edge set E. Let M be a matching of G. We call the matching M *maximal* if there exists no other matching M' such that $M \subset M'$. We call a maximal matching of G that has the largest cardinality a *maximum matching* of G.



(a) Matching. (b) Maximal matching. (c) Maximum matching.

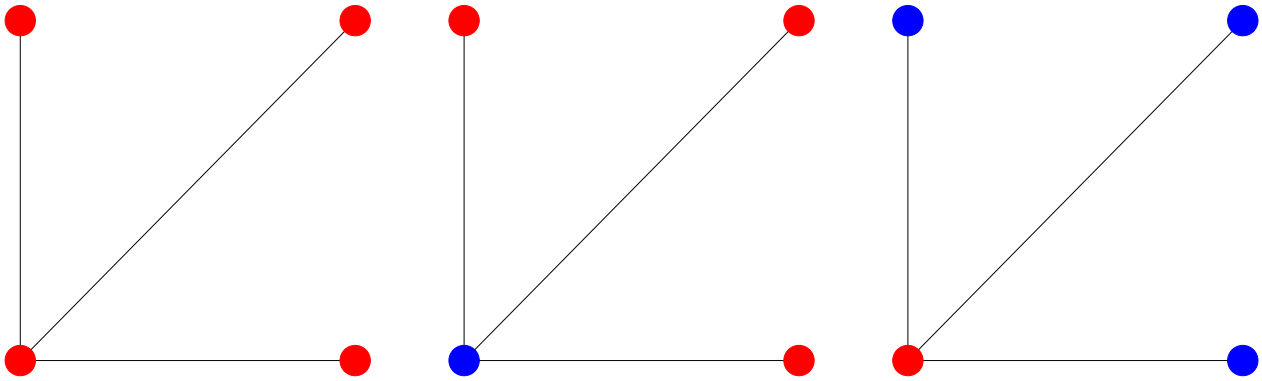
Figure 3.2: Matchings in a graph.

Solving the *Maximum Matching* problem means finding a maximum matching in a graph.

3.2.3 Minimum Vertex Cover problem

Definition 3.2.5 (Vertex cover)**.** Let G be a graph with vertex set V and edge set E. A *vertex cover* is a set $S \subseteq V$ such that for all edges $(u, v) \in E$, we have either $u \in S$ or $v \in S$.

Definition 3.2.6 (Minimal/Minimum vertex cover)**.** Let G be a graph with vertex set V and edge set E. Let S be a vertex cover of G. We call S *minimal* if there is no other vertex cover S' such that $S' \subseteq S$. We call a minimal vertex cover that has smallest cardinality a *minimum vertex cover*.



(a) Vertex cover. (b) Minimal vertex cover. (c) Minimum vertex cover.

Figure 3.3: Vertex covers in a graph.

Solving the *Minimum Vertex Cover* problem means finding a minimum vertex cover in a graph.

3.2.4 Maximum Independent Set problem

Definition 3.2.7 (Independent set)**.** Let G be a graph with vertex set V and edge set E. An independent set $A \subseteq V$ is a set of vertices such that for all $v \in A$ we have $N(v) \cap A = \emptyset$.

Definition 3.2.8 (Maximal/Maximum independent set)**.** Let G be a graph with vertex set V and edge set E. We call an independent set A *maximal* if for all $v \in V$ one of two statements holds.

- $v \in A$, or
- $N(v) \cap A \neq \emptyset$.

We call a maximal independent set of G that has the largest cardinality a *maximum independent set* of G.

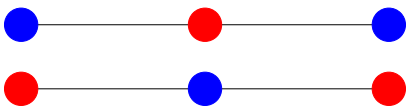


Figure 3.4: Maximal/Maximum independent sets. Note the upper graph indicates a maximal independent set, while the lower graph indicates a maximum independent set.

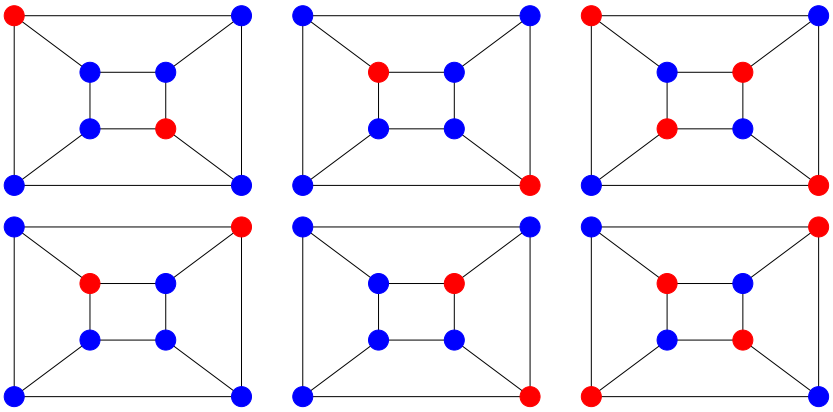


Figure 3.5: Maximal/Maximum independent sets.

Solving the *Maximum Independent Set* problem means finding a maximum independent set in a graph.

3.3 Decision problems

We rigorously define different kinds of decision problems and the link to the optimization problems we discussed in Section 3.2. We use the same notation and reasoning as in [42].

Definition 3.3.1 (Decision problem)**.** A *decision problem* Π is a set of instances $\mathcal{I}$ that can be partitioned into "Yes" and "No" instances $\mathcal{I}_Y, \mathcal{I}_N$, such that

- $\mathcal{I}_Y \cup \mathcal{I}_N = \mathcal{I}$.
- $\mathcal{I}_Y \cap \mathcal{I}_N = \emptyset$.

Example 3.3.2.

1. $\mathcal{I} = \{\{1,4,6\},\{3,11,8\},\{13,2,15\}\}$, $\mathcal{I}_Y = \{A \in \mathcal{I} \ : \ \exists a \in A$ such that a is prime$\}$, $\mathcal{I}_N = \{A \in \mathcal{I} \ : \ \forall a \in A$, a is not prime$\}$.
2. $\mathcal{I} =$ undirected graphs, $\mathcal{I}_Y = \{$graphs with a MIS of size $|MIS| > 10\}$, $\mathcal{I}_N = \{$graphs with a MIS of size $|MIS| \leq 10\}$.

△

Definition 3.3.3 (Certificate)**.** We call x a certificate if it can be used to determine whether $I \in \mathcal{I}_Y$ or $I \in \mathcal{I}_N$.

Example 3.3.4. Certificates for Example 3.3.2(1) could be the following.

$\{1,4,6\}$ has no certificate ,
$\{3,11,8\}$ with certificate 3,
$\{13,2,15\}$ with certificate 13,
$\{13,2,15\}$ with certificate 2.

△

Definition 3.3.5 (NP)**.** We say decision problem Π is in complexity class NP, if every "Yes" instance $I \in \mathcal{I}_Y$ of Π has a certificate x whose validity can be verified in time polynomial in $|I|$.

Definition 3.3.6 (co-NP)**.** We say decision problem Π is in complexity class co-NP, if every "No" instance $I \in \mathcal{I}_N$ of Π has a certificate x whose validity can be verified in time polynomial in $|I|$.

Definition 3.3.7 (Efficient algorithm)**.** Algorithm ALG for a decision problem Π is *efficient* if for every instance I of Π, its computation time is bounded by a polynomial function of the size $|I|$ of I.

Definition 3.3.8 (P)**.** We say decision problem Π is in complexity class P, if there exists an algorithm ALG that determines efficiently for every instance $I \in \mathcal{I}$ of Π, whether $I \in \mathcal{I}_Y$ or $I \in \mathcal{I}_N$.

Some decision problems are harder than others. We say decision problem A is at least as hard as decision problem B if there exists a poly-time reduction from problem B to problem A.

Definition 3.3.9 (Poly-time reduction)**.** A *poly-time reduction* from decision problem Π_1 to decision problem Π_2 is a function $\phi : \mathcal{I}_1 \to \mathcal{I}_2$ that maps an instance $I_1 \in \mathcal{I}_1$ of Π_1 to some instance $I_2 = \phi(I_1) \in \mathcal{I}_2$ of Π_2, so that

- $I_2 = \phi(I_1)$ can be computed in time polynomial in $|I_1|$, for all $I_1 \in \mathcal{I}_1$.
- $I_1 \in (\mathcal{I}_1)_Y \iff I_2 = \phi(I_1) \in (\mathcal{I}_2)_Y$.

If such a poly-time reduction exists, we say Π_1 can be reduced to Π_2, and write $\Pi_1 \preceq \Pi_2$.

Using poly-time reductions, we can make statements about the hardness of a problem. We have the following intuitive result.

Lemma 3.3.10. *Reductions are transitive. If* $\Pi_1 \preceq \Pi_2$ *and* $\Pi_2 \preceq \Pi_3$*, then also* $\Pi_1 \preceq \Pi_3$.

A class of problems that are all equally hard to solve, are the NP-complete problems.

Definition 3.3.11 (NP-complete)**.** A decision problem Π is said to be *NP-complete* if

- $\Pi \in$ NP.
- For all problems $\Pi' \in$ NP, we have $\Pi' \preceq \Pi$.

Note that using the transitivity property of Lemma 3.3.10, we can determine if a decision problem Π is NP-complete by finding a poly-time reduction from a problem that is known to be NP-complete to problem Π. This transitive property also implies that if we can find an efficient algorithm to solve a NP-complete problem, we are able to solve all NP-complete problems.

Using all the definitions we have introduced, we can make the main statement of this section.

Theorem 3.3.12. *A poly-time algorithm to solve only one NP-complete problem* Π*, yields that all problems* $\Pi' \in$ *NP are solvable in polynomial time, hence P=NP.*

3.4 NP-complete problems

There are many problems for which we know they are NP-complete. A list with some of these problems can be found in [17]. The notion of NP-complete problems is introduced by R. Karp. He proved for 21 problems that they are NP-complete. One of these problems is the Maximum Clique problem.

Lemma 3.4.1. *Maximum Clique problem is NP-complete. [27]*

Using this result, we can prove the MIS problem is NP-complete by showing there exists a poly-time reduction from the Maximum Clique problem to the MIS problem.

Lemma 3.4.2. *The MIS problem is NP-complete. [39]*

3.5 Exact algorithms for MIS problem

We show how one can determine the maximum independent set of a graph in a naive way.

Example 3.5.1 (Naive algorithm)**.** We can do the following to determine the maximum independent set of a graph $G = (V, E)$.

Naive algorithm

1. Select a subset $V' \subseteq V$.
2. Check if the subset is a maximal independent set of G.
3. Do this for all possible $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ subsets of V.
4. Select a subset $V' \subseteq V$ that gives the biggest independent set.

By definition, this algorithm gives back a maximum independent set of the graph G. Note this algorithm runs in time $O(2^n \cdot n^2)$. △

The time complexity of the naive algorithm shown is not "good". By considering branch and bound techniques and good bookkeeping, one can reduce this time complexity to $O(1.1996^n n^{O(1)})$ [47]. This is a big improvement, but still, it is exponential in the size of the graph. For small graphs, this is no problem, but for large graphs, this algorithm will never finish.

3.5.1 MIS problem for bipartite graphs

For bipartite graphs, we can solve the MIS problem in polynomial time. We do this by using König's Theorem, introduced by D. König in 1931. An example of König's Theorem is outlined in Figure 3.6.

Theorem 3.5.2 (König's Theorem [15])**.** *In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.*

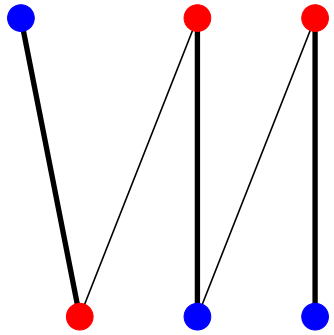


Figure 3.6: A bipartite graph with a minimum vertex cover given by the red vertices and a maximum matching given by the thick edges. Note the cardinality of both sets is the same.

Lemma 3.5.3. *If G is a bipartite graph, a maximum matching can be determined in polynomial time.*

Proof. This is a straightforward application of the Ford-Fulkerson algorithm for maximum flows [21]. □

Lemma 3.5.4. *Let $G = (V, E)$ be a graph. Then S is a vertex cover of G if and only if $V \backslash S$ is an independent set of the graph G.*

Proof. Let $v, w \in V \backslash S$. Assume $(v, w) \in E$. Then there exists an edge that is not incident to any $s \in S$. Hence, S is not a vertex cover. Therefore, we must have $V \backslash S$ is an independent set. Proof in the other direction can be done in the same way. □

Since we can find a minimum vertex cover, we know the quantity $V \backslash S$ is maximized. Therefore, we can find a maximum independent set in polynomial time. Note this does not solve P=NP, since we are only able to determine a maximum independent set for graphs of a certain type. To solve the MIS problem, one needs to find an algorithm that does this for all possible graph types.

3.6 Heuristic algorithms for MIS problem

As we observed, some problems may have no polynomial time algorithm for finding the exact solution. However, in practice, it is often acceptable to solve a problem approximately, rather than exactly. This gave rise to the notion of heuristic algorithms. These are algorithms that solve the problem approximately, but do so in reasonable time. Heuristic algorithms can be used when approximate solutions are sufficient and exact algorithms can not determine the solution in a sufficient amount of time. For example, in the case of NP-hard problems, heuristic algorithms can be a good replacement of exact algorithms.

Definition 3.6.1 (Approximation algorithm)**.** Given a combinatorial optimization problem Π and $\alpha \geq 1$, algorithm *Alg* is an *α-approximation algorithm* for Π if

1. Alg computes a solution for all instances $I \in \mathcal{I}$ in time polynomial in $|I|$.
2. Alg has a performance guarantee α, that is, if $OPT(I)$ is the optimal solution value, then

$$Alg(I) \geq \frac{1}{\alpha} OPT(I), \text{ for all } I \in \mathcal{I}.$$

Example 3.6.2 (Minimum degree greedy algorithm for MIS problem)**.** One of the simplest heuristic algorithms known is the greedy algorithm with minimum degree heuristic.

Greedy algorithm

1. Select one of the vertices that has the smallest degree uniformly at random.
2. Add this vertex to the independent set.
3. Remove all the neighbors of this vertex.
4. Continue doing this till no edges are left.

The set you end up with is a maximal independent set. △

We have the following well-known performance guarantee for the Minimum degree greedy algorithm.

Lemma 3.6.3. *Let G be a graph with maximum degree d_{max}. The Greedy algorithm is an approximation algorithm of the MIS problem with approximation factor $d_{max} + 1$.*

Proof. Let S be the output of the Greedy algorithm and $OPT \subseteq V$ is a maximum independent set of G.

We know each $v \in V \backslash S$ is removed by the algorithm. Removal only happens if you are a neighbor of a vertex $w \in S$. Therefore, we have

$$|V| - |S| = |V \backslash S| \leq d_{\max} |S|.$$

By rewriting, we get

$$|S| \geq \frac{1}{d_{\max} + 1} |V| \geq \frac{1}{d_{\max} + 1} |OPT|.$$

This completes the proof. □

In Figure 3.7, we show how the Greedy algorithm works for an example graph.

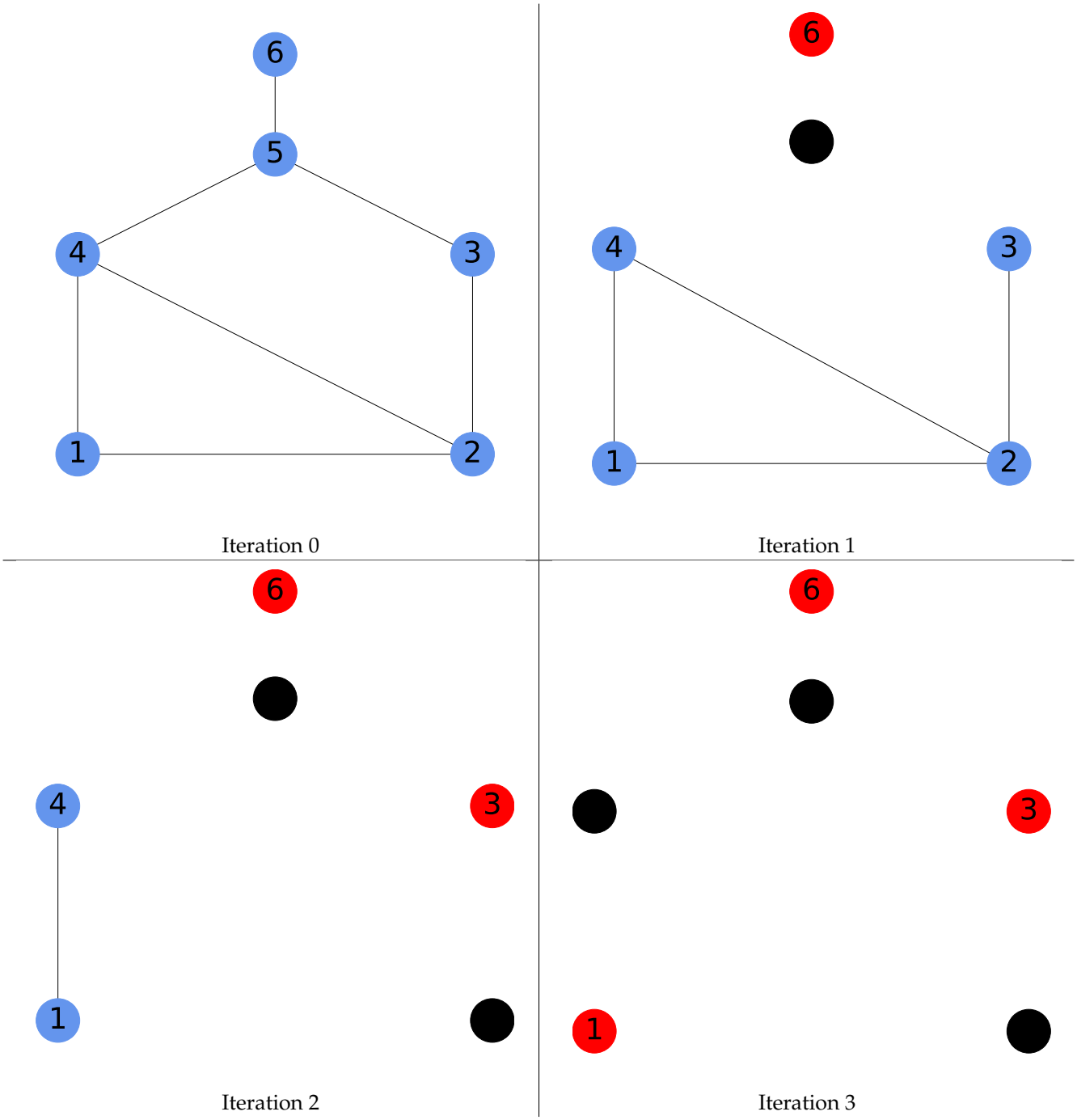


Figure 3.7: Greedy algorithm example. Red nodes indicate the vertex is selected, while black nodes indicate the vertex is removed during the run of the algorithm. The output of the algorithm is the set $\{v_1, v_3, v_6\}$.

In Section 5 and Section 6, we introduce two new heuristic algorithms that generate maximal independent sets in a graph.

4 Mathematical Ecology

This section talks about the generalized Lotka-Volterra equations and properties of the underlying dynamical system. These equations are considered a cornerstone of modern ecology. Aside from ecology, applications for these equations are found in [3, 31].

4.1 Lotka-Volterra equations in 2 dimensions [41, p.156]

The Lotka-Volterra equations are originally posted for two interacting species. Introduced by A. Lotka in 1910 and later V. Volterra in 1926 independently, initially these equations are used to describe the fish catches in the Adriatic Sea.

Definition 4.1.1 (Lotka-Volterra equations in 2 dimensions)**.** Let $x_1 \in \mathbb{R}$ be the number of prey and $x_2 \in \mathbb{R}$ the number of predators. Let $\alpha, \beta, \gamma, \delta$ be positive real parameters describing the interaction between both species. The 2-dimensional Lotka-Volterra equations are defined as

$$\begin{aligned} \frac{dx_1}{dt} &= \alpha x_1 - \beta x_1 x_2, \\ \frac{dx_2}{dt} &= \delta x_1 x_2 - \gamma x_2, \end{aligned} \tag{5}$$

where dx_1/dt and dx_2/dt represent the growth of both populations at time t.

Example 4.1.2 (Fox & rabbit)**.** A well-known application of the Lotka-Volterra equations is given by the modeling of foxes and rabbit populations. Assume we are looking at a nature reserve that is only inhabited by foxes and rabbits. Let's say the foxes multiply at a rate δ (by eating rabbits). The rabbits multiply themselves by rate α (by eating grass). In the absence of rabbits, the foxes die at a rate γ. Rabbits are eaten by foxes at a rate β. Using the Lotka-Volterra equations, one can model this as

$$\begin{aligned} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \left(\begin{pmatrix} \alpha \\ -\gamma \end{pmatrix} + \begin{pmatrix} 0 & -\beta \\ \delta & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right) \\ &= \begin{pmatrix} x_1(\alpha - \beta x_2) \\ x_2(-\gamma + \delta x_1) \end{pmatrix}, \end{aligned}$$

where x_1 represents the number of rabbits and x_2 represents the number of foxes.

Calculating fixed points, we have

$$\begin{aligned} &\text{Fixed point 1: } (0,0). \\ &\text{Fixed point 2: } (\gamma/\delta, \alpha/\beta). \end{aligned}$$

We can calculate the stability of the fixed points by looking at the Jacobian of the system. That is

$$J(x_1, x_2) = \begin{pmatrix} \alpha - \beta x_2 & -\beta x_1 \\ \delta x_2 & -\gamma + \delta x_1 \end{pmatrix}.$$

Evaluated at the fixed points, this becomes

$$J(0,0) = \begin{pmatrix} \alpha & 0 \\ 0 & -\gamma \end{pmatrix}, \qquad J(\gamma/\delta, \alpha/\beta) = \begin{pmatrix} 0 & -\beta\gamma/\delta \\ \delta\alpha/\beta & 0 \end{pmatrix}.$$

This means $(0,0)$ is a saddle point and $(\gamma/\delta, \alpha/\beta)$ is a center. In Figure 4.1 one can see what the trajectories look like.

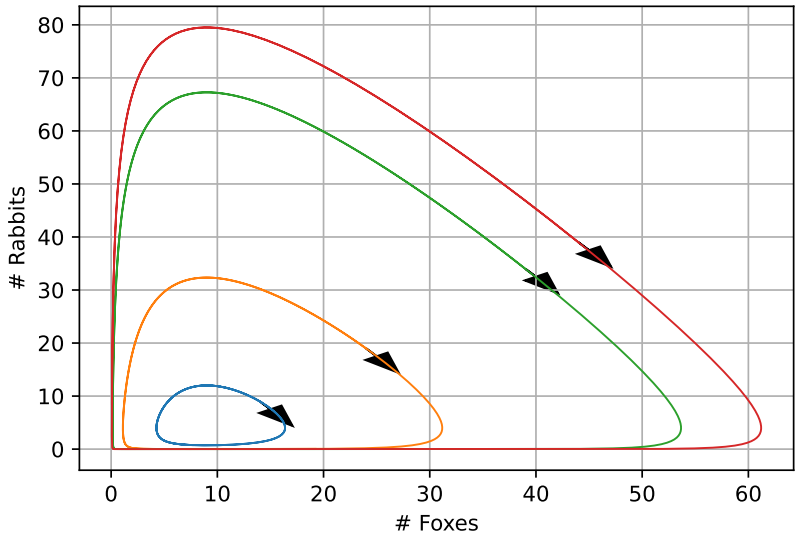


Figure 4.1: Evolution of foxes and rabbit populations under Lotka-Volterra dynamics. Different colors indicate different initial conditions.

Note that this intuitively agrees with reality. When there are less rabbits, the foxes have nothing to eat and hence there are less foxes. When there are less foxes, the rabbits survive more often, and therefore there will be more rabbits. But this results in more foxes, and so on. △

4.2 Lotka-Volterra equations in n dimensions

The n-dimensional Lotka-Volterra equations are a generalization of the 2-dimensional equations we have already seen. We assume there are n interacting species, where every species can possibly interact with every other species.

Definition 4.2.1 (Generalized Lotka-Volterra equations)**.** The generalized n-dimensional Lotka-Volterra equations are given by

$$\frac{dx_i}{dt} = x_i(r_i + (Ax)_i). \tag{6}$$

Where A is a $n \times n$ matrix and r is a vector of real numbers. The matrix A is called the interaction matrix. In vector notation, the generalized Lotka-Volterra equations can be written as

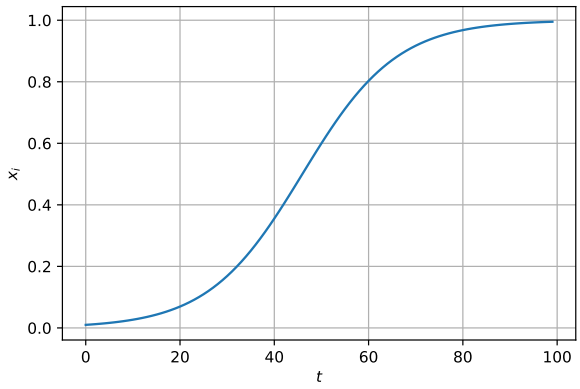
$$\frac{dx}{dt} = \text{diag}(x)(r + Ax). \tag{7}$$

The generalized Lotka-Volterra equations can be used to model the evolution of multiple interacting species. This interaction can be either positive (mutualism) or negative (competition or predation). All these interaction terms can be modeled via the interaction matrix A [11].

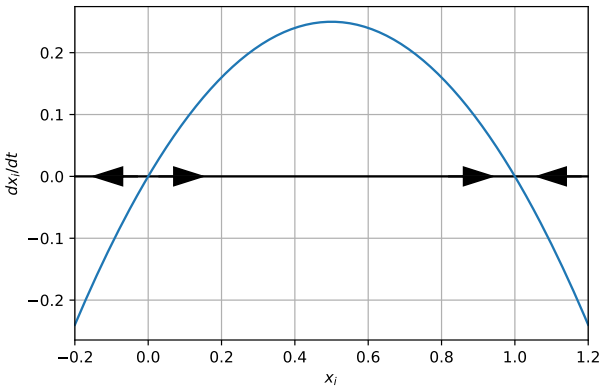
Example 4.2.2 (Logistic equation)**.** What if all the interaction terms are zero? If we look at the equations (6), we see that it simplifies to

$$\frac{dx_i}{dt} = x_i(r_i + A_{ii}x_i).$$

This equation is known as the logistic equation. If $r_i = 1$ and $A_{ii} = -1$, analysis gives the fixed point 0 and 1, where 0 is unstable and 1 is stable. Shown in Figure 4.2 is a corresponding trajectory and the corresponding phase portrait.



(a) Trajectory.



(b) Phase portrait.

Figure 4.2: Logistic equation with $x_0 = 0.1$, $r_i = 1$ and $A_{ii} = -1$.

△

Lemma 4.2.3. *If the initial state x_0 is inside $\mathbb{R}^n_{\geq 0}$, the trajectory $x(t)$, $t \in \mathbb{R}_{\geq 0}$ started at x_0 will be in $\mathbb{R}^n_{\geq 0}$ for all $t \in \mathbb{R}_{\geq 0}$.*

Proof. Note the derivative at the boundary of $\mathbb{R}^n_{\geq 0}$ is for every $i \in [n]$ given by

$$\begin{aligned}\left.\frac{dx_i}{dt}\right|_{x_i=0} &= \left. x_i(r_i + (Ax)_i)\right|_{x_i=0} \\ &= \left. 0 \cdot (r_i + (Ax)_i)\right|_{x_i=0} \\ &= 0.\end{aligned}$$

Therefore, the trajectory can never leave $\mathbb{R}^n_{\geq 0}$. □

In ecological terms, Lemma 4.2.3 tells us the size of a species population can never be negative. This is something one would expect in an ecological system.

4.3 Limiting behavior of Lotka-Volterra equations.

The behavior of the Lotka-Volterra equations entirely depends on the parameter r and the interaction matrix A. M. Hirsch showed that for dimensions bigger or equal than five, all possible limiting behavior can occur [25].

For the 2-dimensional Lotka-Volterra equations, a classification of all possible limiting behavior is given by I. Bomze [8]. This classification is given in terms of the 3-dimensional replicator equations. These equations are equivalent to the Lotka-Volterra equations.

Definition 4.3.1 (Replicator equations)**.** The n-dimensional *Replicator equations* are given by

$$\frac{dx_i}{dt} = x_i((Bx)_i + xBx), \tag{8}$$

where B is a $n \times n$ matrix.

The following result has been proven by I. Bomze [8].

Lemma 4.3.2. *The n-dimensional Lotka-Volterra equations are equivalent to the $n+1$-dimensional replicator equations. That is, for every $A \in \mathbb{R}^{n \times n}$, there exists a matrix $B \in \mathbb{R}^{(n+1)\times(n+1)}$ and a mapping $\mathbb{R}^{n+1} \to \mathbb{R}^n$ such that the mapped trajectories of the replicator equations with matrix B are the same as those of the Lotka-Volterra equations with matrix A.*

Using this lemma, we can determine all the trajectories of the Lotka-Volterra equations by looking at trajectories of the replicator equations. I. Bomze showed all possible limiting behavior of the replicator equations in [8]. Therefore, also all kinds of the limiting behavior of the Lotka-Volterra equations in three dimensions are known.

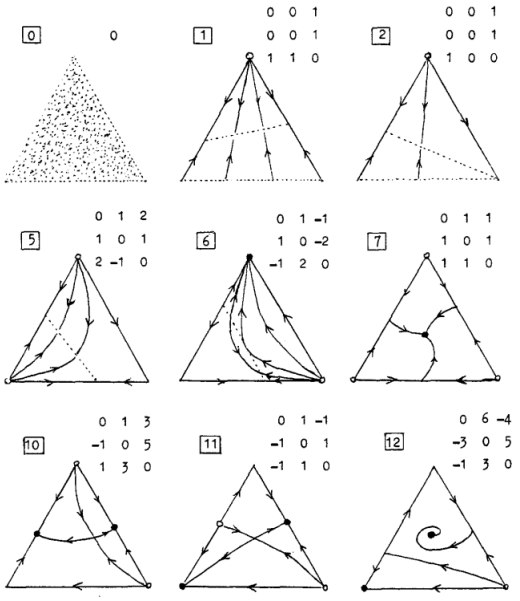


Figure 4.3: Possible limiting behavior of trajectories of the replicator equations [8, p.207]. Note that showing the embedding in 2 dimensions is enough because the replicator equations are normalized as $\sum_{i\in[n]} x_i = 1 - x_{n+1}$.

4.3.1 Fixed points

There is exactly one fixed point of the generalized Lotka-Volterra equations such that $x_i \neq 0$ for all $i \in [n]$. This fixed point x^* must solve the system of equations

$$(r + Ax^*) = 0 \to Ax^* = -r \to x^* = -A^{-1}r.$$

If $x^* > 0$ component-wise, we call x^* feasible.

Aside from this fixed point, there are $2^n - 1$ other fixed points. For every $i \in [n]$, we must have either $x_i^* = 0$ or $x_i^* = (-A^{-1}r)_i$ for x^* to be a fixed point. That is

$$\begin{cases} x_i^* = 0 & \text{or} \\ x_i^* = (-A^{-1}r)_i & \text{for all } i \in \{1, 2, \ldots, n\}. \end{cases} \tag{9}$$

Let x^* be a fixed point of the Lotka-Volterra equations. Let I be the set of indices such that $i \in I$ if $x_i^* = 0$. We can rewrite condition (9) as

$$x_i = (-A'^{-1}r')_i \text{ for all } i \notin I,$$

where A' is the submatrix of the matrix A, obtained by removing rows and columns $i \in I$, and r' is the vector obtained by removing indices $i \in I$ from r.

Example 4.3.3. We look at the Lotka-Volterra system with interaction matrix A and parameter r given by

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \; r = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

One can calculate the fixed points by solving

$$\begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix} = \begin{pmatrix} x_1(1 + x_1 + 2x_2) \\ x_2(1 + 3x_1 + 4x_2) \end{pmatrix}.$$

This gives the fixed points

$$x_1^* = (0,0),\ x_2^* = (0,-1/4),\ x_3^* = (-1,0),\ x_4^* = (1,-1).$$

Note we have $2^2 = 4$ different fixed points. Also, we can see that the unique fixed point given by $x_4^* = -A^{-1}1$ is not feasible since it lies outside the positive quadrant. △

4.3.2 Jacobian matrix

Classification of the stability of fixed points can be done by finding the Jacobian of the system.

Lemma 4.3.4 (Jacobian of Lotka-Volterra equations)**.** *The Jacobian of the Lotka-Volterra system with interaction matrix A and parameter r is given by*

$$J(x) = diag(r) + diag(x)A + diag(Ax). \tag{10}$$

Proof. Assume $i = j$. We calculate

$$\begin{aligned} J_{ii} &= \frac{d}{dx_i}\frac{dx_i}{dt} \\ &= \frac{d}{dx_i}\left(x_i(r_i + (Ax)_i)\right) \\ &= r_i + (Ax)_i + x_i A_{ii}. \end{aligned}$$

Assume $i \neq j$. We calculate

$$\begin{aligned} J_{ij} &= \frac{d}{dx_j}\frac{dx_i}{dt} \\ &= \frac{d}{dx_j}\left(x_i(r_i + (Ax)_i)\right) \\ &= x_i\frac{d}{dx_j}(r_i + (Ax)_i) \\ &= x_i A_{ij}. \end{aligned}$$

In clean notation, this becomes

$$J(x) = \text{diag}(r) + \text{diag}(x)A + \text{diag}(Ax).$$

□

We have already seen that there is a fixed point given by $x^* = -A^{-1}r$, the Jacobian evaluated at this point has a special form.

Lemma 4.3.5 (Jacobian of feasible fixed point)**.** *Let A be invertible. The Jacobian of the Lotka-Volterra system with interaction matrix A and parameter r, evaluated at x^*, is given by*

$$J(x^*) = diag(-A^{-1}r)A. \tag{11}$$

Proof. We can plug in the fixed point $x^* = -A^{-1}r$ into the equation derived in Lemma 4.3.4.

$$\begin{aligned} J(A^{-1}r) &= \text{diag}(r) + \text{diag}(-A^{-1}r)A + \text{diag}(-AA^{-1}r) \\ &= \text{diag}(r) + \text{diag}(-A^{-1}r)A - \text{diag}(r) \\ &= \text{diag}(-A^{-1}r)A. \end{aligned}$$

This completes the proof. □

5 Lotka-Volterra equations for solving MIS problem

In Section 3 and Section 4, we introduced the MIS problem and the Lotka-Volterra equations. At first sight, these subjects do not seem to have much in common. It turns out however, that we can use the Lotka-Volterra equations to find maximal independent sets in a graph. One can do this by choosing appropriate parameters. In this section we discuss how to do this. We prove the Lotka-Volterra equations indeed converge to a maximal independent set.

The idea of using a set of differential equations to solve a combinatorial optimization problem has been used before. In 1996, Bomze et al [9] showed one can use the replicator equations we discussed in Section 4.3 to determine a maximal clique in a graph. As shown in Section 3.4, we can transform a maximal clique to a maximal independent set by looking at the complement graph. In this section we show, independently from [9] and based on different principles, that the Lotka-Volterra equations can be used to directly determine this maximal independent set.

First, we state the main theorem of this section and we spend time proving it. Finally, we discuss the implications of this theorem more specifically for regular graphs and complete graphs.

The main theorem of this section states we can use the adjacency matrix of a graph, together with the Lotka-Volterra equations to find a maximal independent set in the graph.

Theorem 5.0.1* (Main Theorem)**.** *Let $G = (V, E)$ be a simple graph of size n with adjacency matrix A^0. Let $\tau > 1$ and $A = -(\tau A^0 + \mathbb{I})$. Let $x(t)$ be the trajectory of the Lotka-Volterra equations (Section 4.2) with interaction matrix A, parameter $r = \mathbf{1}$ and initial condition $x_0 \in (0,1)^n$. Then $x^* := \lim_{t\to\infty} x(t)$ exists and the set $\{v_i \in V \ : \ x_i^* = 1\}$ is a maximal independent set of G.*

Note that this statement depends on a parameter $\tau > 1$. We see later there exists other bounds for τ if we only consider certain types of graphs. To simplify notation, we assume that x_0 is not a fixed point in the rest of this section.

5.1 Ecological heuristic

The Lotka-Volterra equations have ecological meaning, as discussed in Section 4. By taking the interaction matrix $A = -(\tau A^0 + \mathbb{I})$, we implicitly define an ecological system.

This ecological system consists of n species, associated to the vertices of the graph, that all have the same negative effect on the neighboring species, given by the interaction strength τ. One can compare this situation to an ecological system consisting of different species fighting for the same resources. Figure 5.1 shows the corresponding ecological food web.

Figure 5.1: Three species fighting for the same resources with negative interaction strength τ.

The interaction matrix also has values minus one on the diagonal. This represents the negative effect a species has on itself. If the size of a species grows too much, the maximum food capacity is reached and there will not be enough food for everyone. Since we only consider undirected graphs, we consider ecological systems where species X has an effect on species Y when species Y also has an effect on species X.

Theorem 5.0.1* tells us that if we let this ecological system go on for a long time, every species either dies out, or ends up in a state without any competition.

5.2 Matrix Theory

We first need some definitions and lemmas to help us with the proof.

Definition 5.2.1 (Positive/Negative definite)**.** A $n \times n$ hermitian matrix A is called negative definite if $x^*Ax < 0$ for all nonzero complex vectors $x \in \mathbb{C}^n$. In the case of a real matrix A, this condition simplifies to $x^TAx < 0$ for all nonzero real vectors $x \in \mathbb{R}^n$. A positive definite matrix is defined as $x^*Ax > 0$ for all nonzero complex vectors $x \in \mathbb{C}^n$ and if A is real $x^TAx > 0$ for all nonzero real vectors $x \in \mathbb{R}^n$.

Lemma 5.2.2. *For a real symmetric matrix A and diagonal matrix D with positive diagonal elements, the eigenvalues of DA are the same as those of $D^{1/2}AD^{1/2}$, where $D^{1/2}$ fulfills the property $D^{1/2}D^{1/2} = D$.*

Proof. We can rewrite

$$\begin{aligned}\det(DA - \lambda\mathbb{I}) &= \det(D^{1/2}[D^{1/2}AD^{1/2} - \lambda\mathbb{I}]D^{-1/2}) \\ &= \det(D^{1/2}) \cdot \det(D^{1/2}AD^{1/2} - \lambda\mathbb{I}) \cdot \det(D^{-1/2}).\end{aligned}$$

So solving $\det(DA - \lambda\mathbb{I}) = 0$ has the same solutions as solving $\det(D^{1/2}AD^{1/2} - \lambda\mathbb{I}) = 0$. Hence, DA and $D^{1/2}AD^{1/2}$ have the same eigenvalues. □

To prove a matrix is not negative definite, we later introduce Sylvester's criterion. To use this theorem, we need some prerequisites concerning matrix minors.

Definition 5.2.3 (Minor)**.** Let A be a $m \times n$ matrix and k an integer with $0 < k < m, n$. A $k \times k$ minor of A is the $k \times k$ matrix obtained from A by deleting $m - k$ rows and $n - k$ columns.

Definition 5.2.4 (Principal minor)**.** If rows and columns with the same indices remain unremoved, we call the minor a principal minor.

Definition 5.2.5 (Leading principal minor)**.** If a principal minor is a square upper-left submatrix of the larger matrix, the principal minor is called a leading principal minor. For a $n \times n$ square matrix, there are n leading principal minors.

Example 5.2.6. We show all the minors of the matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$.

1×1 minors	2×2 minors	3×3 minors
$(1)^*$	$\begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}^*$
(2)	$\begin{pmatrix} 4 & 6 \\ 7 & 9 \end{pmatrix}$	
(3)	$\begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix}$	
(4)	$\begin{pmatrix} 2 & 3 \\ 8 & 9 \end{pmatrix}$	
(5)	$\begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix}$	
(6)	$\begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix}$	
(7)	$\begin{pmatrix} 2 & 3 \\ 5 & 6 \end{pmatrix}$	
(8)	$\begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}$	
(9)	$\begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}^*$	

Figure 5.2: Minors of a matrix. Principal minors are colored red. Leading principal minors are denoted with an asterisk.

Note the principal minors are a subset of all minors. In the same way, the leading principal minors are a subset of all principal minors. △

Lemma 5.2.7 (Sylvester's criterion [19]). *An $n \times n$ Hermitian matrix A is positive definite if and only if all of the leading principal minors have positive determinant.*

Lemma 5.2.8 (Adapted Sylvester's criterion). *An $n \times n$ Hermitian matrix A is negative definite if and only if the determinant of the leading principal minor of size k has sign $(-1)^k$.*

Proof. Assume A is negative definite. Then $-A$ is positive definite. For every principal leading minor d_k^{-A} of size k of the matrix $-A$, we know $\det(d_k^{-A}) > 0$. Note the leading principal minor of size k of A is given by $d_k^A = -d_k^{-A}$. Calculating the determinant gives

$$\det(d_k^A) = \det(-d_k^{-A}) = (-1)^k \det(d_k^{-A}).$$

This expression is positive when k is even and negative when k is odd.

Proof in the other direction can be done in the same way. □

5.3 Stability of the feasible fixed point

In this section, we use n to denote the number of Lotka-Volterra equations and the size of a graph. We have seen in Section 4.3.1 that the n-dimensional Lotka-Volterra equations generally have 2^n fixed points. Not all these points are interesting if we consider the ecological context of the system. The following lemma shows that for appropriate initial conditions, we can discard a lot of fixed points.

Lemma 5.3.1* (Prison Lemma). *Let $x(t)$, $t \in \mathbb{R}_{\geq 0}$ denote the trajectory of the n-dimensional Lotka-Volterra equations with $A = -(\tau A^0 + \mathbb{I})$ and $\mathbf{r} = 1$. Let x_0 denote the initial condition of the system. If $x_0 \in [0,1]^n$, then $x(t) \in [0,1]^n$ for all $t \in \mathbb{R}$.*

Proof. We know by Lemma 4.2.3 the trajectory will never leave $\mathbb{R}^n_{\geq 0}$. Let $i \in [n]$ and $x \in [0,1]^n$. Then

$$\begin{aligned}
\left.\frac{dx_i}{dt}\right|_{x_i=1} &= \left. x_i[1 - (Ax)_i] \right|_{x_i=1} \\
&= \left. x_i - x_i A_{ii} x_i - x_i \tau \sum_{j \neq i} A^0_{ij} x_j \right|_{x_i=1} \\
&= 1 - 1 - \tau \sum_{j \neq i} A^0_{ij} x_j \\
&= -\tau \sum_{j \neq i} A^0_{ij} x_j \leq 0.
\end{aligned}$$

So, if the trajectory starts inside the unit hypercube, it stays inside the unit hypercube. □

Because of Lemma 5.3.1*, we know we can never reach a fixed point x^* if $x^* \notin [0,1]^n$. Therefore, if we assume that $x_0 \in (0,1)^n$, we can assume without loss of generality that x^* is inside the unit hypercube. First we assume that x^* satisfies $Ax^* = \mathbf{1}$ and $0 < x^* < 1$ component-wise. We will determine the Jacobian evaluated at the point x^*. We show that when $\tau > 1$, this Jacobian has an eigenvalue λ such that $Re(\lambda)$ is bigger than zero. From this, we can conclude x^* is an unstable fixed point [37].

Lemma 5.3.2*. *Let $\tau > 1$. If $0 < x^*_k < 1$ for some $k \in [n]$, then there exists some $v_l \in N(v_k)$ such that $0 < x^*_l < 1$. By $N(v_k)$ we denote the neighborhood of a vertex v_k as discussed in Section 1.*

Proof. We know x^*_k obeys the relation $(Ax^*)_k = 1$. Rewriting it gives

$$\begin{aligned}
(Ax^*)_k &= \sum_{j \in [n]} A_{kj} x^*_j \\
&= x^*_k + \tau \sum_{j \neq k} A^0_{kj} x^*_j \\
&= x^*_k + \tau \sum_{j : v_j \in N(v_k)} x^*_j.
\end{aligned}$$

Since $0 < x_k^* < 1$, we know $0 < \tau \sum_{j:v_j \in N(v_k)} x_j^* < 1$. Because $\tau > 1$, we must have $0 < x_l^* < 1$ for some $l \in \{i : v_i \in N(v_k)\}$. □

Lemma 5.3.3⋆. *Let x^* satisfy $Ax^* = \mathbf{1}$ and $0 < x^* < 1$ component-wise. For $\tau > 1$, the Jacobian evaluated at x^* has an eigenvalue with positive real part. Hence, x^* is an unstable fixed point.*

Proof. By Lemma 4.3.4, we know the Jacobian is given by

$$J(x^*) = \operatorname{diag}(x^*)A.$$

As shown in Lemma 5.2.2, we know the spectrum of $J(x^*)$ coincides with the spectrum of

$$M(x^*) = \operatorname{diag}(x^*)^{1/2} A \operatorname{diag}(x^*)^{1/2}.$$

Since this is a symmetric and real matrix, we know by Lemma 1.3.2 that all eigenvalues of $J(x^*)$ are real.

For $i \neq j$ we can write out the Jacobian elements explicitly as

$$J(x^*)_{ii} = -x_i^* \text{ and } J(x^*)_{ij} = -\tau A_{ij}^0 x_i^*.$$

By assumption, $0 < x_k^* < 1$ for some $k \in [n]$. Hence, by Lemma 5.3.2⋆ there exists some $l \in \{i : v_i \in N(v_k)\}$, such that $x_l^* \in (0,1)$. We can rewrite the Jacobian so that row/column k becomes the first row/column and row/column l becomes the second row/column. Without loss of generality we can assume $k = 1$ and $l = 2$. This gives the matrix elements

$$\begin{aligned} J(x^*)_{11} &= -x_1^*, \\ J(x^*)_{22} &= -x_2^*, \\ J(x^*)_{12} &= -\tau x_1^*, \\ J(x^*)_{21} &= -\tau x_2^*. \end{aligned}$$

We use the Adapted Sylvester's criterion on the hermitian matrix $M = \operatorname{diag}(x^*)^{1/2} A \operatorname{diag}(x^*)^{1/2}$. Note

$$\begin{aligned} M(x^*)_{11} &= -x_1^*, \\ M(x^*)_{22} &= -x_2^*, \\ M(x^*)_{12} &= -\tau {x_1^*}^{1/2} {x_2^*}^{1/2}, \\ M(x^*)_{21} &= -\tau {x_1^*}^{1/2} {x_2^*}^{1/2}. \end{aligned}$$

The determinant of the 2×2 leading principal minor of M is given by

$$M_{11}M_{22} - M_{12}M_{21} = x_1^* x_2^* - \tau^2 x_1^* x_2^*.$$

By the Adapted Sylvester's criterion we know M is not negative definite if $x_1^* x_2^* - \tau^2 x_1^* x_2^* < 0$. Solving gives $\tau > 1$.

Therefore, we know $J(x^*)$ must have a positive eigenvalue. Hence, x^* is unstable. □

Lemma 5.3.4. *The point $x^* = \lim_{t\to\infty} x(t)$ exists for every initial condition $x_0 \in (0,1)^n$ and lies on a corner of the unit hypercube.*

Proof. In the proof of Theorem 5.3.3⋆, we showed that all eigenvalues of $J(x^*)$ are real if x^* obeys $Ax^* = \mathbf{1}$. Since all eigenvalues are real and there exists an eigenvalue with positive real part, we know x^* must be hyperbolic and unstable. Therefore, x^* must be on the corner of the unit hypercube. Since we can never leave the unit hypercube if we start inside it, we know oscillatory behavior around x^* can not occur. Therefore, we know $x^* = \lim_{t\to\infty} x(t)$ exists and is a point on the corner of the unit hypercube. □

5.4 Stability of fixed points with extinction

Before we have seen that for τ large enough, the fixed point x^* given by $Ax^* = \mathbf{1}$ is unstable if it is not on a corner of the unit hypercube. We have seen in Section 4.3.1 that the Lotka-Volterra equations have 2^n different fixed points, because for every $i \in [n]$, we have either

$$x_i^* = 0 \text{ or } (Ax^*)_i = 1.$$

Recall that we have shown in Lemma 4.3.4 that the Jacobian of the system is given by

$$J(x) = \mathbb{I} - \text{diag}(x)A - \text{diag}(Ax).$$

Let $x \in [0,1]^n$ be a point inside the unit hypercube. Define $I = \{i \in [n] \ : \ x_i = 0\}$. We can write the Jacobian evaluated at x in a special way.

- For $i \in I, j \notin I$;

$$J(x)_{ij} = 0.$$

- For $i \in I, j \in I$;

$$\begin{aligned} J(x)_{ij} &= \mathbb{1}_{\{i=j\}} - \mathbb{1}_{\{i=j\}} \sum_{k=1}^{n} A_{ik}x_k \\ &= \mathbb{1}_{\{i=j\}}(1 - \sum_{k=1}^{n} A_{ik}x_k). \end{aligned}$$

- For $i \notin I, j \notin I$;

$$\begin{aligned} J(x)_{ij} &= \mathbb{1}_{\{i=j\}} - x_i A_{ij} - \mathbb{1}_{\{i=j\}} \sum_{k=1}^{n} A_{ik}x_k \\ &= \mathbb{1}_{\{i=j\}} - x_i A_{ij} - \mathbb{1}_{\{i=j\}} \sum_{k \notin I} A_{ik}x_k \\ &= J'(x). \end{aligned}$$

  We denote by A' the matrix A where we remove all rows and columns corresponding to indices in I. In the same way, J' corresponds to the resulting Jacobian.

- For $i \notin I, j \in I$;

$$\begin{aligned} J(x) &= \mathbb{1}_{\{i=j\}} - x_i A_{ij} - \mathbb{1}_{\{i=j\}} \sum_{k=1}^{n} A_{ik}x_k \\ &= -x_i A_{ij}. \end{aligned}$$

We can reorder our terms so that $x_i = 0$ for $i \in \{l+1, \ldots, n\}$, where $l = n - |I|$. We can thus rewrite the Jacobian as

$$J(x) = \begin{pmatrix} J' & [-x_i A_{ij}]_{i,j \notin I} \\ 0 & \text{diag}(1 - \sum_{k=1}^{n} A_{ik}x_k) \end{pmatrix}. \tag{12}$$

Lemma 5.4.1*. *Let $x^* := \lim_{t \to \infty} x(t)$ be a fixed point of the system described in Theorem 5.0.1*. If x^* is on a corner of the unit hypercube, then for every $i \in [n]$ such that $x_i^* = 0$, there exists a index $k \in \{j \ : v_j \in N(v_i)\}$ such that $x_k^* = 1$.*

Proof. Let $I = \{i \in [n] \ : \ x_i^* = 0\}$. We know the Jacobian is given by Equation 12.

To determine stability of the fixed point, we want to determine the eigenvalues of the Jacobian evaluated at the fixed point.

$$\begin{aligned}
\det(J_{ij}(x^*) - \lambda \mathbb{I}) &= \det \begin{pmatrix} J' - \lambda \mathbb{I} & -x_i^* A_{ij} \\ 0 & \operatorname{diag}(1 - \sum_{k=1}^{n} A_{ik} x_k^* - \lambda) \end{pmatrix} \\
&= \det(J' - \lambda \mathbb{I}) \det(\operatorname{diag}(1 - \sum_{k=1}^{n} A_{ik} x_k^* - \lambda)) \\
&= \det(J' - \lambda \mathbb{I}) \prod_{i=l+1}^{n} (1 - \sum_{k=1}^{n} A_{ik} x_k^* - \lambda).
\end{aligned}$$

Therefore, we know eigenvalues λ_i, $i \in \{l+1, dots, n\} = I$, are given by

$$\begin{aligned}
\lambda_i &= 1 - \sum_{k=1}^{n} A^{ik} x_k^* \\
&= 1 - x_i^* - \tau \sum_{k \neq i} A_{ik}^0 x_k^* \\
&= 1 - \tau \sum_{k \neq i} A_{ik}^0 x_k^* \\
&= 1 - \tau \sum_{k : v_k \in N(v_i)} x_k^*.
\end{aligned}$$

For every $i \in I$, we see there exists an eigenvalue given by

$$\lambda_i = 1 - \tau \sum_{k : v_k \in N(v_i)} x_k^*.$$

Assume that for some $v_i \in V$, we have $\{k \ : \ v_k \in N(v_i)\} \subseteq I$. This gives

$$\lambda_i = 1 - 0 = 1.$$

So in this case x^* can never be a stable fixed point. Therefore, we must have $\{k \ : \ v_k \in N(v_i)\} \not\subset I$. Hence, there exists some $v_k \in N(v_i)$ such that $x_k^* > 0$. Since x^* lies on a corner of the unit hypercube, we know that $x_k^* = 1$. □

Lemma 5.4.2*. *If $x^* := \lim_{t \to \infty} x(t)$ is on the corner of the unit hypercube, then the support of x^* corresponds to an independent set in the underlying graph.*

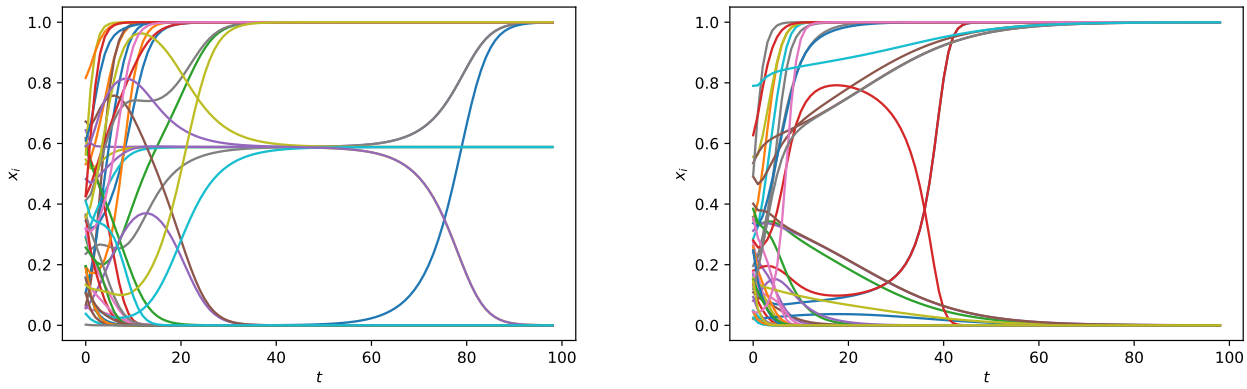
Proof. We know that for every $i \in [n]$, x_i^* is either equal to zero or has to obey the equation $(Ax^*)_i = 1$. Let $k \in [n]$ be such that $x_k^* = 1$. Rewriting gives

$$\begin{aligned}
(Ax^*)_k &= \sum_{j \in [n]} A_{kj} x_j^* \\
&= x_k^* + \tau \sum_{j \neq k} A_{kj}^0 x_j^* \\
&= 1 + \tau \sum_{j : v_j \in N(v_k)} x_j^*.
\end{aligned}$$

Since this has to equal one, we know $x_j^* = 0$ for all $j \in \{i \ : \ v_i \in N(v_k)\}$. Hence, $\{v_i \ : \ x_i^* = 1\}$ is an independent set in the underlying graph. □

For $\tau > 1$, it follows from Lemma 5.3.3* that $\lim_{t\to\infty} x(t)$ is a point on a corner of the unit hypercube. Therefore, Lemma 5.4.2* gives that the corresponding set in the graph is an independent set. Using Lemma 5.4.1*, we know this set is also a maximal independent set. This completes the proof of Main Theorem 5.0.1*.

In Figure 5.3, we have plotted trajectories of the Lotka-Volterra equations. For τ equal to 0.7, we see the trajectories do not converge to a corner of the unit hypercube. For τ equal to 1.1 we see that the trajectories do converge to a corner of the unit hypercube. We have just proven that the set of variables that converge to one correspond to a maximal independent set in the underlying graph.



(a) Interaction parameter $\tau = 0.7$.

(b) Interaction parameter $\tau = 1.1$.

Figure 5.3: Trajectories of the Lotka-Volterra equations with $A = -(\tau A^0 + \mathbb{I})$, $r = \mathbf{1}$ and random initial condition $x_0 \in (0,1)^n$. The matrix A^0 is the adjacency matrix of a connected graph of size $n = 40$.

5.5 Regular Graphs

In the proof of Main Theorem 5.0.1*, we showed that $\tau > 1$ is a lower bound for convergence of the system to a maximal independent set. In the case of regular graphs, we are able to specify this bound more clearly.

5.5.1 Fixed points

In Lemma 1.6.1, we have already seen that for regular graphs, the vector $\mathbf{1}$ is an eigenvector of the adjacency matrix. Using this, we can specify the interior fixed point $x^* = -A^{-1}\mathbf{1}$.

Lemma 5.5.1*. *Let G be a regular graph with constant degree d and adjacency matrix A^0. The feasible fixed point of the Lotka-Volterra equations with $\tau > 0$, $A = -(\tau A^0 + \mathbb{I})$ and $r = \mathbf{1}$ is given by*

$$x_i^* = \frac{1}{\tau d + 1} \text{ for all } i \in [n]. \tag{13}$$

Proof. We know from Lemma 1.6.1 that A^0 has eigenvalue d corresponding to eigenvector $\mathbf{1}$.

Note if λ is an eigenvalue of A, then λ^{-1} is an eigenvalue of A^{-1}. Since $A = -(\tau A^0 + \mathbb{I})$, we know $-(\tau d + 1)$ is an eigenvalue of A. Therefore, $\frac{-1}{\tau d+1}$ is an eigenvalue of A^{-1} with eigenvector $\mathbf{1}$. So we have

$$-A^{-1}\mathbf{1} = \frac{1}{\tau d + 1}\mathbf{1}.$$

This completes the proof. □

5.5.2 Stability

We have already shown in Lemma 4.3.5 that the Jacobian of the Lotka-Volterra equations, evaluated at $x^* = -A^{-1}\mathbf{1}$, is given by

$$J(x^*) = \operatorname{diag}(x^*)A.$$

In the case of a regular graph, we know by Lemma 5.5.1* that x^* is a constant vector. This gives rise to the following lemma.

Lemma 5.5.2* (Jacobian of regular graph)**.** *Let G be a regular graph with constant degree d. Let λ be an eigenvalue in the spectrum of G. Then the value*

$$T(\lambda) = -\frac{\tau\lambda + 1}{\tau d + 1} \tag{14}$$

is an eigenvalue of $J(x^*)$. Where $J(x^*)$ is the Jacobian of the Lotka-Volterra equations evaluated at $x^* = A^{-1}\mathbf{1}$.

Proof. Let λ be an eigenvalue of A^0 with eigenvector v. Note

$$\begin{aligned} J(x^*)v &= \operatorname{diag}(x^*)Av \\ &= \operatorname{diag}\left(\frac{1}{\tau d + 1}\right) Av \\ &= -\operatorname{diag}\left(\frac{1}{\tau d + 1}\right)(\tau\lambda + 1)v \\ &= -\frac{\tau\lambda + 1}{\tau d + 1} v. \end{aligned}$$

□

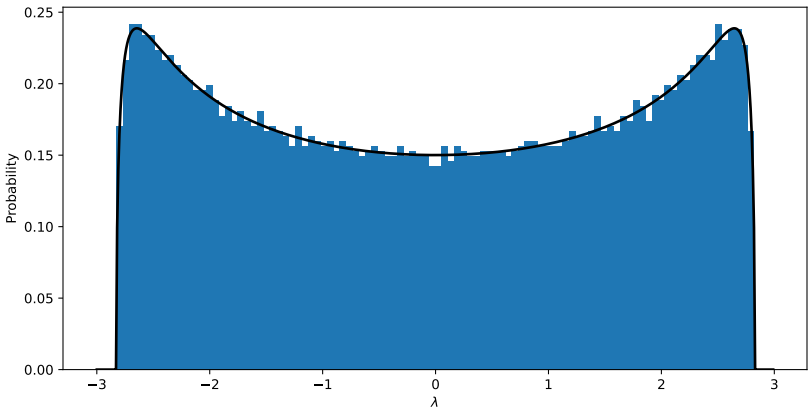


Figure 5.4: Density plot of the spectrum of a 3-regular graph of size 5000. The black line is a plot of the McKay distribution for degree 3.

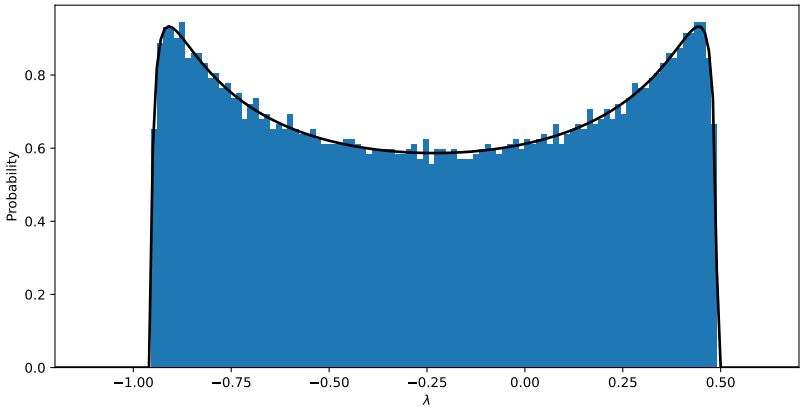


Figure 5.5: Density plot of the spectrum of the Jacobian evaluated at the feasible fixed point x^* given by $Ax^* = \mathbf{1}$. The black line is a plot of the transformed McKay distribution for degree 3.

It follows from the McKay distribution properties (Section 2.2) that almost surely every eigenvalue λ of a d-regular graph obeys the bound

$$-2\sqrt{d-1} \leq \lambda \leq d.$$

Using the transformation above, we know the eigenvalues of $J(x^*)$ almost surely obey the bound

$$T(d) \leq \lambda \leq T(-2\sqrt{d-1})$$
$$-1 \leq \lambda \leq -\frac{-2\tau\sqrt{d-1}+1}{\tau d+1}.$$

So, for $\tau < \frac{1}{2\sqrt{d-1}}$, the interior fixed point is almost always stable. For $\frac{1}{2\sqrt{d-1}} < \tau < 1$, one can use this expression to calculate the probability that the interior fixed point is stable.

5.6 Complete Graphs

For complete graphs we can show the lower bound $\tau > 1$ is necessary for convergence to a maximal independent set. That is, for every $\tau \leq 1$, this convergence will not occur.

5.6.1 Fixed points

Note that every complete graph is also a regular graph. Hence, the interior fixed point is given by

$$x^* = \frac{1}{\tau d + 1} = \frac{1}{\tau(n-1)+1}. \tag{15}$$

5.6.2 Stability

To determine stability of the interior fixed point, we again use the Adapted Sylvester's criterion (Lemma 5.2.8). We also use the following lemma.

Lemma 5.6.1 (Determinant complete graph)**.** *Let the matrix $A \in \mathbb{R}^{n\times n}$ be given by*

$$A_{ij} = \begin{cases} a & \text{if } i = j. \\ b & \text{if } i \neq j. \end{cases}$$

Then $\det(A) = (a + (n-1)b)(a-b)^{n-1}$.

Proof. Note the determinant of a matrix is invariant under row manipulation. Using the column operation $C_1 \leftarrow \sum_{j=1}^{n} C_j$, we get a matrix with $a-(n-1)b$ on the first column. Note

$$\begin{pmatrix} a+(n-1)b & b & \dots & b \\ a+(n-1)b & a & \dots & b \\ a+(n-1)b & b & \dots & b \\ \vdots & \vdots & \ddots & \vdots \\ a+(n-1)b & b & \dots & a \end{pmatrix} = \begin{pmatrix} 1 & b & b & \dots & b \\ 1 & a & b & \dots & b \\ 1 & b & a & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & b & \dots & \dots & a \end{pmatrix} \begin{pmatrix} a+(n-1)b & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

So we have

$$\begin{vmatrix} a+(n-1)b & b & \dots & b \\ a+(n-1)b & a & \dots & b \\ a+(n-1)b & b & \dots & b \\ \vdots & \vdots & & \vdots \\ a+(n-1)b & b & \dots & a \end{vmatrix} = (a+(n-1)b) \begin{vmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & a-b & 0 & \dots & 0 \\ \vdots & 0 & a-b & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & 0 & \dots & \dots & a-b \end{vmatrix}$$
$$= (a+(n-1)b)(a-b)^{n-1}.$$

This completes the proof. □

Just as in the proof of Main Theorem 5.0.1*, we can use the Adapted Sylvester's criterion to determine stability of the feasible fixed point. Note for a complete graph we have

$$J(x^*) = \begin{pmatrix} -x & -\tau x & \dots & -\tau x \\ -\tau x & -x & & \vdots \\ \vdots & & \ddots & \vdots \\ -\tau x & \dots & \dots & -x \end{pmatrix},$$

where x is given in Equation 15.

So using Lemma 5.6.1, we can calculate the determinant

$$\begin{aligned} \det(J(x^*)) &= (-x + (n-1)\cdot -\tau x)(-x+\tau x)^{n-1} \\ &= x^n(-1)^n(1+\tau(n-1))(1-\tau)^{n-1} \\ &= \left(\frac{1}{\tau(n-1)+1}\right)^n (-1)^n(1+\tau(n-1))(1-\tau)^{n-1}. \end{aligned}$$

We can distinguish between the following cases.

- $\tau = 1$. In this case we know the determinant is always zero. Hence, we can not make conclusions on the stability of the fixed point.
- n odd, $\tau \neq 0$. In this case, the determinant is always smaller than zero if $\tau > 0$.
- n even, $\tau < 1$. In this case, the determinant is bigger than zero.
- n even, $\tau > 1$. In this case, the determinant is smaller than zero.

We see that $\tau = 1$ is a bifurcation point of the system. So for complete graphs, we know $\tau = 1$ is a strict lower bound for the Lotka-Volterra equations to converge to a maximal independent set.

6 Numerical Continuation Algorithm

In Section 5, we showed that for τ larger than one, the LV equations can be used to find maximal independent sets in a graph. To solve the MIS problem, we need to find a maximal independent set with maximum cardinality. Since there can be many different fixed points of the LV equations, we can use these equations to find multiple maximal independent sets in a graph. To find the maximum independent set in a graph, it is key to find the basin of attraction of all the fixed points that correspond to a maximum independent set.

One possible way of finding this basin of attraction is by trying many different initial states. Doing this could possibly result in a maximum independent set, but we can never be sure if this is indeed the largest possible maximal independent set. Trying many initial states is also computationally expensive for large graphs. This is because for a graph of size $n \in \mathbb{N}$, the initial state is in the domain $(0,1)^n$. This domain grows exponentially in the size of the graph.

In this section, we discuss a modification to the algorithm introduced in Section 5. This modification avoids the initial value problem we just encountered. This happens at the cost of computation time. The algorithm we introduce here still does not solve the MIS problem, but numerically it performs better than the algorithm introduced in Section 5.

6.1 Numerical continuation

The algorithm we propose is inspired by the concept of numerical continuation. We discuss this concept here.

Numerical continuation describes the behavior of fixed points under the change of parameters in the underlying dynamical system. The easiest way to illustrate this is by an example.

Example 6.1.1 (Pitchfork bifurcation)**.** We look at the dynamical system given by the ordinary differential equation

$$\frac{dx}{dt} = rx - x^3,$$

where $r, x \in \mathbb{R}$. Determining fixed points gives

$$x^* = 0 \text{ if } r < 0, \text{ and}$$
$$x^* = 0,\ x^* = \pm\sqrt{r} \text{ if } r \geq 0.$$

The bifurcation diagram is drawn in Figure 6.1.

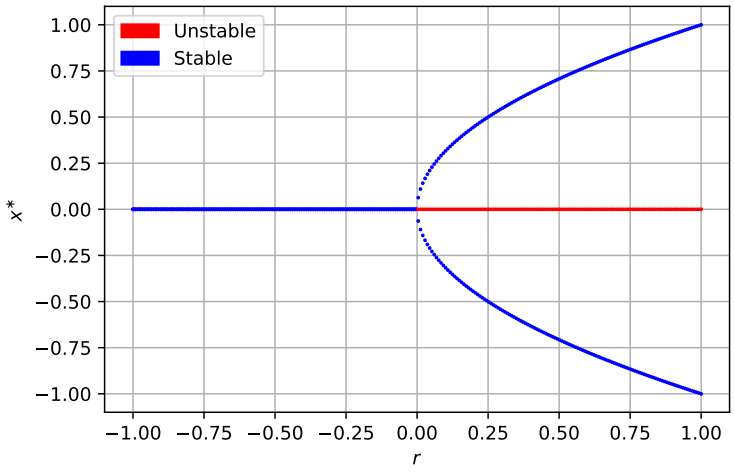


Figure 6.1: Bifurcation diagram of the function $\frac{dx}{dt} = rx - x^3$.

We see it depends on the parameter r whether we have one or three fixed points. For $r > 0$, we conclude from this analysis that if $x_0 > 0$, $x(t)$ converges to $\sqrt{r}$. But if $x_0 < 0$, $x(t)$ converges to $-\sqrt{r}$.

We see that when we follow the fixed points from $r = -1$ till $r = 1$, we have two possible end states. Thus we know how the fixed point(s) change under perturbation of the parameter space.

△

6.2 Algorithm

Note that for τ equal to zero we have $A^{-1}\mathbf{1} = \mathbf{1}$. Inspired by the numerical continuation, we propose the following algorithm.

Numerical continuation algorithm

1. Input: graph $G = (V, E)$ with $V = (v_i)_{i\in[n]}$ and parameter $\tau_{\text{step}} \in \mathbb{R}_{>0}$.
2. Let $\tau = 0$ and $x^* = (1, 1, \ldots, 1)$.
3. Increase τ by τ_{step} and run the Lotka-Volterra system for long time with $A = -(\tau A^0 + \mathbb{I})$, $r = \mathbf{1}$ and $x_0 = x^*$. Denote the fixed point obtained by x^*.
4. Keep returning to step 3 until $\tau > 1$.
5. Return all vertices v_i such that $x_i = 1$.

Note this algorithm is following the behavior of the fixed point obtained for $x_0 = (1, 1, \ldots, 1)$. So instead of only increasing the time parameter in the system, we also slowly increase the interaction parameter τ in the system.

One of the difficulties with this algorithm is the step size τ_{step}. We do not know when the stability of $A^{-1}1$ will change in advance. Therefore, for this algorithm to perform properly, we need to use a very small step size. But a small step size results in a computationally expensive algorithm.

Numerically, the behavior of the fixed point is shown in Figure 6.2.

6.3 Ecological heuristic

This modified algorithm has ecological meaning. Instead of letting the system run for some $\tau > 1$, as we have done in Section 5, we now add extra information by taking into account all $\tau < 1$.

Just as in Section 5, we are looking at an ecological system with negative interaction strength τ and self-limiting rate one. The difference is we are now looking at τ for which a species goes extinct. So instead of giving species an unfair chance of survival by choosing certain initial coordinates, we now give every species a fair chance of survival.

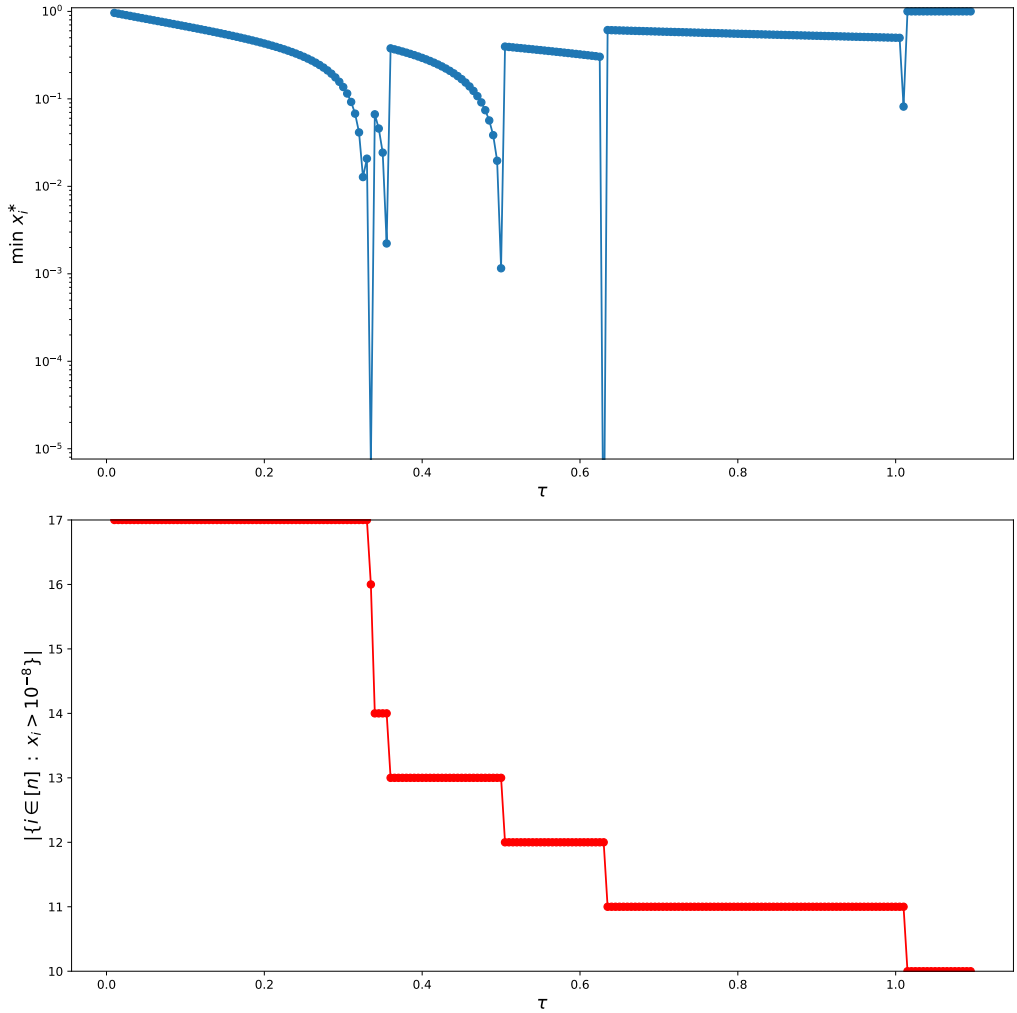


Figure 6.2: The minimum fixed point coordinate plotted against the interaction parameter τ. Only coordinates that are bigger than 10^{-8} are taken into account. We have used a connected graph of size $n = 17$. We see that for some values of τ, at least one of the coordinates of x^* converges to zero. One can interpret this as a species going extinct. On the lower plot, we have plotted the number of variables that are bigger than 10^{-8}. We see that we end up with a maximal independent set of size 10. We used a step size of $\tau_{\text{step}} = 0.005$.Note that for $\tau > 1$, all coordinates of the fixed point are either zero or one. This is also what we have shown in Section 5.

It remains to determine the values of τ for which variables are converging to zero.

6.4 Bifurcation values

The algorithm depends on the nearest value τ^* such that the stability of the Jacobian evaluated at $-A^{-1}\mathbf{1}$ flips from stable to unstable. The hard part of the algorithm is finding these bifurcation values.

The Jacobian satisfies

$$J(x^*) = \text{diag}(x^*)A.$$

The bifurcation happens exactly when $\lambda_{\max}(J(x^*)) = 0$. This happens when $\det(J(x^*)) = 0$. Rewriting gives $\det(\text{diag}(x^*)A) = \det(\text{diag}(x^*))\det(A)$. This expression can be zero only in two cases.

1. $\det(A) = 0$, or
2. $-A^{-1}1 \notin (0,1)^n$.

Case 1 happens when $\det(\tau A^0 + \mathbb{I}) = 0$. This happens exactly when either

- $\tau = \frac{-1}{\lambda}$ for $\lambda > 0$ in the spectrum of A^0, or
- $\tau = \frac{1}{|\lambda|}$ for $\lambda < 0$ in the spectrum of A^0.

Note that A^0 always has positive and negative eigenvalues (Lemma 1.3.3).

Since we only consider positive values of τ, the minimum τ for which $\det(A) = 0$ is given by

$$\tau^* = \frac{1}{|\lambda_{\min}(A^0)|}. \tag{16}$$

6.4.1 Newton's method

For the second case where $x^* = A^{-1}\mathbf{1}$ has a coordinate that equals zero, we implement Newton's method. We use Newton's method with the functional

$$\begin{aligned} F : \mathbb{R}^n &\to \mathbb{R} \\ x^* &\mapsto \prod_{i\in[n]} x_i^*. \end{aligned}$$

Note that F equals zero exactly when one of the coordinates of x^* equals zero. We can calculate the derivative of this function as follows.

$$\begin{aligned} \frac{dF}{d\tau} &= \sum_{j\in[n]} \frac{\partial x_j^*}{\partial \tau} \frac{\partial}{\partial x_j^*} \prod_{i\in[n]} x_i^* \\ &= \sum_{j\in[n]} \frac{\partial x_j^*}{\partial \tau} \prod_{i\neq j} x_i^*. \end{aligned}$$

We can calculate $\frac{\partial}{\partial \tau} x^*$ as follows.

$$\frac{\partial}{\partial \tau} x^* = \frac{\partial}{\partial \tau} A^{-1} 1.$$

Note that for a matrix $B(t)$, we have

$$0 = \frac{\partial}{\partial t} B(t) B(t)^{-1} = \left(\frac{\partial}{\partial t} B(t)\right) B(t)^{-1} + B(t) \left(\frac{\partial}{\partial t} B(t)^{-1}\right).$$

Therefore,

$$\frac{\partial}{\partial t} B(t)^{-1} = -B(t)^{-1} \left(\frac{\partial}{\partial t} B(t)\right) B(t)^{-1}.$$

For the matrix $A(\tau)$, this implies

$$\frac{\partial}{\partial \tau} A^{-1} = -A^{-1} A^0 A^{-1}.$$

So we know

$$\frac{\partial}{\partial \tau} x^* = \frac{\partial}{\partial \tau} A^{-1} 1 = -A^{-1} A^0 A^{-1} 1.$$

Combining everything, we have

$$\frac{dF}{d\tau} = \sum_{j\in[n]} (-A^{-1}A^0A^{-1}1)_j \prod_{i\neq j} x_i^*.$$

With this derivative, we can use the Newton iteration procedure

$$\tau_{n+1} = \tau_n - \frac{F(\tau_n)}{\frac{d}{d\tau}F(\tau_n)}.$$

We know the fixed point of this recursive equation satisfies $F(\tau_n) = 0$.

Conjecture 6.4.1. *The sequence $(\tau_n)_{n\in\mathbb{N}}$ converges monotonically to τ^* such that $F(\tau^*) = 0$.*

6.5 Algorithm speed up

We are going to reduce the computation time of the algorithm by skipping over the domain of τ where no bifurcation occurs. The following theorem tells us we can do this.

Theorem 6.5.1 ([20, p.138])**.** *If the nontrivial equilibrium $x^* = A^{-1}\mathbf{1}$ of the Lotka-Volterra equations is feasible and there exists a constant positive diagonal matrix C such that $CA + A^TC$ is negative definite, then the Lotka-Volterra model is globally stable in the feasible region.*

We know that on the domain D where no bifurcation has occurred, we have for $\tau \in D$ both:

- $\tau < \frac{1}{|\lambda_{\min}(A^0)|}$ (by Equation 16).
- $x^* = A^{-1}\mathbf{1}$ is feasible, that is $x^* \in (0,1)^n$.

By the first condition, we know $\lambda_{\max}(A) < 0$, so A is negative definite and therefore also $A + A^T$ is negative definite. The second condition tells us that we are in the feasible region. By Theorem 6.5.1, we know $A^{-1}\mathbf{1}$ is globally stable. Therefore, we can skip the integration steps where no bifurcation occurs.

6.6 Modified algorithm

We are only able to determine τ^* for the first bifurcation. The way to do this for the next bifurcation is by looking at the graph with the bifurcated vertices removed.

Algorithm 1 NumericalContinuationAlgorithm$(G, \epsilon_{\text{forward}}, \epsilon_{\text{backward}})$

Input: Graph G with vertex set $V = (v_i)_{i\in\{1,2,\dots,n\}}$, edge set E and adjacency matrix A^0. Displacement parameters $\epsilon_{\text{forward}}, \epsilon_{\text{backward}} > 0$.
Output: Maximal independent set of G

1: $x_{\text{end}} \leftarrow (1,1,\dots,1)$.
2: **while** $|E| > 0$ **do**
3: $\quad\tau^* \leftarrow \inf\{\tau > 0 \,:\, \lambda_{\max}(J((\tau A^0 + \mathbb{I})^{-1}1)) = 0\}$.
4: $\quad x_{\text{start}} \leftarrow \lim_{t\to\infty} x(t)$, where $x(t)$ is the trajectory of the LV equations with parameter $\tau_{\text{start}} = \tau^* - \epsilon_{\text{backward}}$, $A = -(\tau_{\text{start}}A^0 + \mathbb{I})$, $r = \mathbf{1}$ and initial condition x_{end}.
5: $\quad x_{\text{end}} \leftarrow \lim_{t\to\infty} x(t)$, where $x(t)$ is the trajectory of the LV equations with parameters $\tau_{\text{end}} = \tau^* + \epsilon_{\text{forward}}$, $A = -(\tau_{\text{end}}A^0 + \mathbb{I})$, $r = \mathbf{1}$ and initial condition x_{start}.
6: $\quad$Remove all vertices v_i such that $(x_{\text{end}})_i = 0$ from the graph G.
7: **end while**
8: **return** All vertices of G that have not converged to zero.

6.7 Algorithm with exceptions

There are some boundary cases for which Algorithm 1 does not perform correctly. We will discuss them here.

6.7.1 Regular graphs

In the case the graph is d-regular and we run the algorithm, no bifurcation will occur. We have already seen that the feasible fixed point of a regular graph is given by

$$x^* = \frac{1}{\tau d + 1}\mathbf{1}. \tag{17}$$

If the initial state consists of a vector with all the same coordinates $c \in (0, 1)$, then the dynamics are given by

$$\frac{dx}{dt} = \text{diag}(c)(\mathbf{1} - (\tau d + 1)c).$$

Hence, the trajectories will converge to constant $x_i^* = \frac{1}{\tau d+1}\mathbf{1}$ and no bifurcation will occur.

6.7.2 Other graphs

There also exists other graphs for which the fixed point remains feasible for all $\tau < 1$. An example is the graph given in Figure 6.3

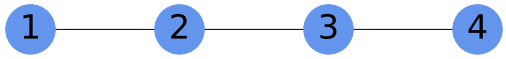


Figure 6.3: Line graph with four vertices.

These exceptions occur when the bifurcation happens due to $\det(A)$ becoming zero. If we have a symmetry in our graph, the algorithm can not choose which variable has to become zero. Therefore, we introduce a perturbation to the system when this happens. That is, we check if $\tau^* = \frac{1}{|\lambda_{\min}(A^0)|}$, and if so, we add a normally distributed number to x_{start} such that we do not leave the unit hypercube. Based on the many simulations we have done, we formulate the following conjecture.

Conjecture 6.7.1. *These exceptions only occur when* $\det(A) = 0$.

6.8 Exception induced by algorithm

The Continuation algorithm has an exception for which it does not work directly. An example of such an exception is given by the graph in Figure 6.4.

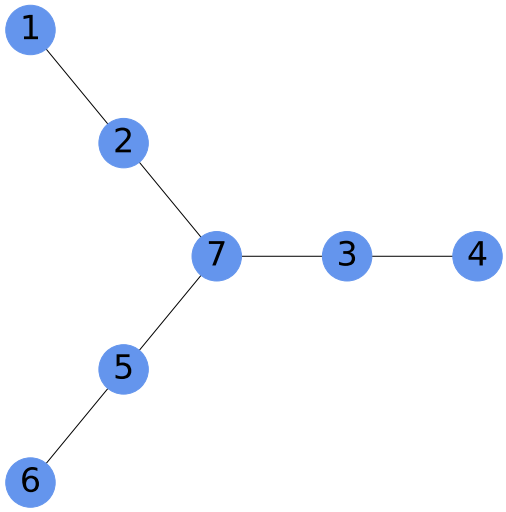


Figure 6.4: Example of exception graph.

The variable corresponding to vertex v_7 will be the first one that will reach zero. Note that once a variable reaches zero, it never leaves zero. Therefore, we end up with the graph in Figure 6.5.

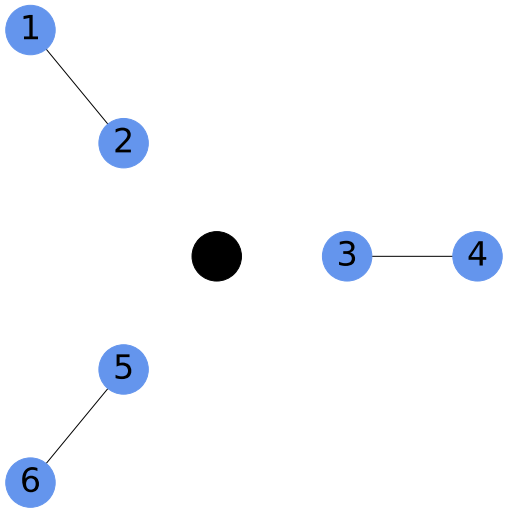


Figure 6.5: Example of exception graph after the first time a variable reaches zero.

In this case we have three disconnected regular graphs. To determine which variables will converge to zero, we need to introduce a small perturbation. It could theoretically be that this results in the variables v_2, v_3 and v_5 converging to zero. In this case we end up with the graph in Figure 6.6.

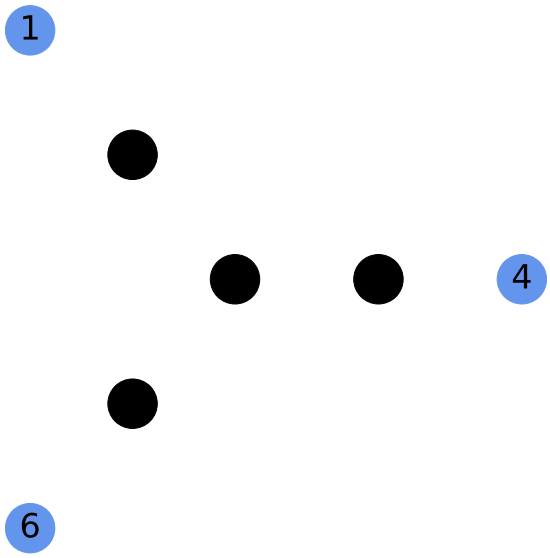


Figure 6.6: Example of exception graph after the algorithm is finished.

Note the algorithm does not return a maximal independent set since we can still add vertex v_1 to the output. We can solve this problem easily in the following way.

- Let I be the set of vertices that the algorithm gives as output.
- For every vertex $v \in I$, remove v from the graph as well as all the neighbors of v.
- Run the algorithm again on the resulting graph if it is not empty.
- Keep doing this until we get a maximal independent set as output.

Theorem 6.8.1. *The continuation algorithm returns a maximal independent set.*

Proof. Note the algorithm is removing vertices from the graph G. Since we keep on removing vertices until the graph is disconnected, we know the algorithm returns an independent set.

If the graph is not empty, we know the algorithm will remove at least one vertex. Therefore, if the exception above does occur, we can run the algorithm again to get a graph with less vertices than before. At some point we end up with an empty graph and we are done. □

6.9 Final algorithm

Here we give a final description of the algorithm. Note that we have taken into account all the discussed exceptions.

Algorithm 2 NumericalContinuationAlgorithm_Iteration(G, $\epsilon_{\text{forward}}$, $\epsilon_{\text{backward}}$)

Input: Graph G with vertex set $V = (v_i)_{i\in\{1,2,\dots,n\}}$, edge set E and adjacency matrix A^0. Displacement parameters $\epsilon_{\text{forward}}, \epsilon_{\text{backward}} > 0$.
Output: Maximal independent set of G

$x_{\text{end}} \leftarrow (1, 1, \dots, 1)$.
while $|E| > 0$ **do**
  $\tau^* \leftarrow \inf\{\tau > 0 \,:\, \lambda_{\max}(J((\tau A^0 + \mathbb{I})^{-1}1)) = 0\}$.
  $x_{\text{start}} \leftarrow \lim_{t\to\infty} x(t)$, where $x(t)$ is the trajectory of the LV equations with parameter $\tau_{\text{start}} = \tau^* - \epsilon_{\text{backward}}$, $A = -(\tau_{\text{start}} A^0 + \mathbb{I})$, $r = \mathbf{1}$ and initial condition x_{end}.
  if $\tau^* = \frac{1}{|\lambda_{\min}(A^0)|}$ **then**
    $x_{\text{start}} = x_{\text{start}} + e$, where $e_i \sim N(0, 10^{-8})$.
  end if
  $x_{\text{end}} \leftarrow \lim_{t\to\infty} x(t)$, where $x(t)$ is the trajectory of the LV equations with parameters $\tau_{\text{end}} = \tau^* + \epsilon_{\text{forward}}$, $A = -(\tau_{\text{end}} A^0 + \mathbb{I})$, $r = \mathbf{1}$ and initial condition x_{start}.
  Remove all vertices v_i such that $(x_{\text{end}})_i = 0$ from the graph G.
end while
$J \leftarrow$ vertices of G that have not converged to zero.
return J

Algorithm 3 NumericalContinuationAlgorithm(G, $\epsilon_{\text{forward}}$, $\epsilon_{\text{backward}}$)

Input: Graph G with vertex set $V = (v_i)_{i\in\{1,2,\dots,n\}}$ and edge set E. Displacement parameters $\epsilon_{\text{forward}}, \epsilon_{\text{backward}} > 0$.
Output: Maximal independent set of G

output = {}.
while output is not a MIS **do**
  G′ ← G.
  for v in output **do**
    Remove $N(v)$ from G'.
    Remove v from G'.
  end for
  output′ ← NumericalContinuationAlgorithm_Iteration(G′, $\epsilon_{\text{forward}}$, $\epsilon_{\text{backward}}$).
  Add all v in output′ to output.
end while
return output.

6.10 Lower bound bifurcation

We show a lower bound for τ until a new bifurcation happens. We will use this bound in the algorithm to improve the computation time of Newton's method. We have the following definitions.

Definition 6.10.1 (Diagonal stability)**.** A real matrix B is diagonally stable if there exists a positive diagonal matrix D such that $DB + B^T D$ has only negative eigenvalues.

Definition 6.10.2 (D-stability)**.** A real matrix B is D-stable if DB has only negative eigenvalues for any diagonal matrix D with strictly positive diagonal entries.

Lemma 6.10.3. *Negative definiteness of a symmetric matrix implies diagonal stability.*

Proof. If B is negative definite, we know it has only negative eigenvalues. By taking D equal to the identity matrix, we conclude $DB + B^T D$ has only negative eigenvalues. □

Lemma 6.10.4 ([28, p.32])**.** *Diagonal stability implies D-stability.*

Therefore, we know if A is negative definite, then $J(x^*) = \text{diag}(x^*)A$ has only negative eigenvalues.

Definition 6.10.5 (Diagonal dominance)**.** A matrix B is strictly diagonal dominant if for every row i of the matrix B we have

$$\sum_{j \neq i} B_{ij} < B_{ii}.$$

Lemma 6.10.6. *If a symmetric and real matrix B is diagonally dominant, it is positive definite.*

Proof. Since B is symmetric and real, we know all the eigenvalues are real. Application of Gershgorin Circle Theorem (1.4.2) tells us every eigenvalue is bigger than zero. Therefore, B is positive definite. □

Taking $\tau < 1/d_{max}$ gives A strictly diagonally dominant. Therefore, A is positive definite. Hence, $-A$ is negative definite and $J(x^*)$ has only negative eigenvalues. So after every bifurcation we know the next bifurcation will not happen before $\tau = 1/d_{max}$.

7 Numerical results

We have done numerical simulations of both algorithms to determine the performance. We do these simulations for the LV algorithm from Section 5, for the Continuation algorithm from Section 6 and for the Minimal degree greedy algorithm we discussed in Section 3.6. We use the last algorithm as a baseline to compare our results. We have run the simulations with $\epsilon_{\text{forward}} = 0.05$. Let $1/d_{\max}$ be the lower bound discussed in Section 6.10 and τ_{-1} the last bifurcation value. We have taken $\epsilon_{\text{backward}}$ equal to $\epsilon_{\text{backward}} = \tau^* - \max\{1/d_{\max}, \tau_{-1}\}/10$.

Source code is available on Github (https://github.com/NiekMooij) or upon request.

We compare the following measures.

Average performance
The average performance is determined by averaging the performance over all simulations. The performance of a simulation is determined by dividing the output of the algorithm by the output of the exact algorithm.

Percentage correct
The percentage correct is determined by dividing the number of times a simulation results in a maximum independent set by the total number of simulations.

Worst-case performance
The worst-case performance is determined by taking the lowest performance from all simulations.

7.1 Erdős-Rényi graphs

For Erdős-Rényi graphs, determining the maximum independent set is an NP-complete problem. We want to compare performance of the algorithms and therefore we need to determine the exact solution. We do this by implementing an algorithm that returns a maximum independent set of the graph. Since this is computationally expensive, we can only do this for small graphs.

We showed in Section 2.1.1 that there exists a critical threshold for which the Erdős-Rényi graphs become almost surely connected. This critical threshold is given by

$$p_{\text{critical}} = \frac{\log n}{n}.$$

To make sure the graphs we use are almost always connected, we use $p = p_{\text{critical}} + 0.1$. Results of the simulations can be found in Appendix A.1 and Appendix A.4.

We see that for Erdős-Rényi graphs, the continuation algorithm on average works better than the LV algorithm and the Greedy algorithm. Also the percentage of graphs for which we got a maximum independent set is the largest for the continuation algorithm. Note that this comes at the price of computation time. If one looks at the average computation time, we see the Continuation algorithm takes much more time than the other two algorithms.

7.2 Geometric graphs

For geometric graphs, determining the maximum independent set is also an NP-complete problem. We want to compare performance of the algorithms and therefore we need to determine the exact solution. We do this by implementing the same exact algorithm as we used for the Erdős-Rényi graphs.

We showed in Section 2.1.2 that there exists a critical threshold for which the generated graphs become almost surely connected. This critical threshold is given by

$$p_{\text{critical}} = \sqrt{\frac{\log n}{\pi n}}.$$

To make sure the graphs we use are almost always connected, we use $p = p_{\text{critical}} + 0.1$. Results of the simulations can be found in Appendix A.2 and Appendix A.4.

For geometric graphs we see the Greedy algorithm has the best average performance, as well as the best worst-case performance. The Greedy algorithm also takes the least time. Note the Continuation algorithm does perform better than the LV algorithm.

7.3 Bipartite graphs

We also determine the performance of the LV algorithm, the Continuation algorithm and the Greedy algorithm on random bipartite graphs. The reason we do this on bipartite graphs is because in this case the maximum independent set can be determined in polynomial time, as shown in Section 3.5.1. This means we can do simulations for bigger graphs.

We also encounter the problem that generating random graphs is hard. For our simulations, we want the graph to be connected, because otherwise, determining a maximal independent set can be done on disconnected subgraphs separately. There is no smart way to do this for general random bipartite graphs. Therefore, we use exhaustive search for connected bipartite graphs of the required size. This means we keep on generating bipartite graphs of the required size until we have generated a graph that is connected.

In the simulations, we use bipartite graphs of size n, where the components of the graph have sizes $0.6n$ and $0.4n$. We use connection probability $p = 0.1$. Results of the simulations can be found in Appendix A.3 and Appendix A.4.

In the case of bipartite graphs, we see the Continuation algorithm works the best. In most cases, it finds a maximum independent set. The Continuation algorithm performs best with respect to the worst-case measure and average performance measure.

7.4 Computational complexity

Results on the computational complexity of the simulations is shown in Appendix A.1/A.2/A.3. We see that the Greedy algorithm performs the fastest for all graph types. The LV algorithm performs the second fastest, while the Continuation algorithm performs by far the slowest.

7.5 Numerical continuation failure

The results of the numerical simulations on the Continuation algorithm still have some cases where the output is not a maximal independent set. Numerically, this happens only when the trajectories of the LV equations do not converge fast enough. We already tried to solve this by changing the integration time and the forward displacement parameter, but we were not able to solve all of it.

We had the following number of failures for different graph types.

- Erdős-Rényi graphs: Failure on 17 of the 6500 runs.
- Geometric graphs: Failure on 54 of the 6500 runs.
- Bipartite graphs: Failure on 1 of the 900 runs.

Discussion

D.1 Results

In this thesis, we proved the Lotka-Volterra equations with parameters $A = -(\tau A^0 + \mathbb{I})$ and $r = \mathbf{1}$ converge to a maximal independent set. For the Continuation algorithm, we showed that up to some exceptions, the algorithm will also return a maximal independent set.

Numerical results
We have done numerical experiments to determine the performance of the algorithms. For Erdős-Rényi graphs and geometric random graphs, the Continuation algorithm has a better performance compared to the LV algorithm. When compared to the Greedy algorithm however, it turned out that the Continuation algorithm does not perform better in general. On Erdős-Rényi graphs the Continuation algorithm performs slightly better, while on geometric random graphs the Greedy algorithm performs slightly better. When we consider bipartite graphs, the Continuation algorithm performs really well compared to the LV algorithm and the Greedy algorithm. This may indicate that the continuation algorithm can not perform well when there are triangles in the graph. This is also what we see with performance on the random geometric graphs.

Biological interpretation
We have already shown in this thesis that we can interpret the Lotka-Volterra equations as a competitive ecological system where species have a negative effect on each other and the size of every species population is bounded. In the Continuation algorithm, we make use of a root finding algorithm that finds the smallest τ such that a variable converges to zero. In ecology this corresponds to a species dying out. Therefore, we can use the techniques implemented in this algorithm to make statements about ecological networks that correspond to the parameters we use. If one is able to abstract this interaction strength from the system, one can make statements about how close a species is to dying out.

Another interesting phenomenon for ecologists is that of reversibility. When a species dies out, one wants to know if it is possible to put this species back from an external source without the species directly dying out. Using the root-finding technique used in Section 6.4.1, one can determine how much the interaction needs to change before a species can be put back in the ecological system.

D.2 Future work

One way of improving the LV algorithm is by finding "good" initial conditions. In this thesis, we took the center of the unit hypercube as the initial condition, to make sure we do not prefer one vertex over the other. In reality a lot of vertices can already be labeled bad by looking at the vertex degree. In a maximum independent set there are mostly vertices that have low degree. Therefore, we could penalize vertices with a high degree by choosing appropriate initial conditions. One possible way of doing this would be choosing

$$(x_0)_i = 1 - \frac{d_i}{d_{\max} - 1} \text{ for } i \in [n].$$

In this way, the vertices with high degree will have small initial values, while vertices with small degree will have relatively large initial values.

The results of the numerical work on the continuation algorithm still have some cases where the output is not a maximal independent set. Numerically, this happens only when the trajectories of the Lotka-Volterra equations do not converge fast enough. We already tried to solve this by changing the integration time and the forward displacement parameter, but we were not able to solve all of it. For future work it is a good idea to do research on the integration time of the algorithm.

To investigate the performance of the Continuation algorithm on graphs with many triangles, one could test the algorithm on graphs generated by replacing the vertices in a graph with triangles. In this way we end up with a graph consisting of triangles "glued" together.

Instead of comparing the output of the algorithms with an exact solution, one can also compare the algorithm to each other directly. In this way we can avoid computing the exact solution, which is the bottleneck in the execution of the code since it takes a lot more time than the approximation algorithms. If we do this we can compare algorithm results for bigger graphs.

A suggestion for future work is to look at random matrices. Instead of taking a matrix with Bernoulli distributed entries as we have implicitly done when we take an Erdős-Rényi graph, one can also let these entries have a more general distribution. For example one can take these entries normally distributed. Proving convergence for this setting would have much broader biological implications.

Another suggestion for future work is to look at graphs that are not as uniform as Erdős-Rényi and geometric graphs. For example, one can look at graphs generated by the block model.

In this thesis we have formulated the following conjectures which we believe are true, but have not been able to prove. They are both inspired by numerical experiments.

Conjecture 7.5.1. *The Newton method described in Section 6.4.1 converges monotonically to* τ^*.

Conjecture 7.5.2. *The exceptions described in Section 6.7 occur only when* $det(A) = 0$.

D.3 Acknowledgements

First of all I want to thank my supervisor Ivan Kryven. He has provided excellent guidance while still giving me the opportunity to keep on failing until I got it right.

I also like to thank the second readers Palina Salanevich and Carla Groenland for taking the time to read this research report.

Last but not least, I want to thank my family and friends for all their support. I could not have done this without all your help. It is my research, but it feels like our accomplishment.

Appendix

A.1 Erdős-Rényi graphs data

Average Performance						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 10	0.9831	0.05989	0.9992	0.01264	0.9956	0.02934
n = 12	0.9864	0.04648	0.9967	0.02304	0.9935	0.0326
n = 14	0.9876	0.04254	0.9956	0.02497	0.9918	0.03369
n = 16	0.986	0.04027	0.9942	0.0264	0.9908	0.03421
n = 18	0.9857	0.0403	0.9927	0.02819	0.9909	0.03346
n = 20	0.9869	0.03496	0.9947	0.02281	0.9933	0.02615
n = 22	0.9853	0.0358	0.9926	0.02542	0.9901	0.03081
n = 24	0.9871	0.03179	0.9935	0.02346	0.9918	0.02742
n = 26	0.9865	0.0317	0.9911	0.0257	0.9887	0.02977
n = 28	0.9869	0.02936	0.992	0.02433	0.9882	0.02843
n = 30	0.9833	0.03045	0.9909	0.02495	0.9878	0.03004
n = 32	0.9829	0.03149	0.9894	0.02492	0.9885	0.02733
n = 34	0.9826	0.03021	0.9909	0.02312	0.9843	0.02965

Average Running Time (10^{-3} s)						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 10	66.8	31.4	702.0	368.0	0.496	0.531
n = 12	65.2	20.6	818.0	275.0	0.596	0.62
n = 14	58.0	7.84	886.0	221.0	0.642	0.488
n = 16	57.4	6.65	1050.0	271.0	0.768	0.462
n = 18	66.7	25.2	1420.0	509.0	1.01	0.598
n = 20	66.8	22.9	1620.0	582.0	1.16	0.614
n = 22	68.3	13.0	1790.0	420.0	1.28	0.512
n = 24	65.9	9.06	1920.0	436.0	1.45	0.602
n = 26	74.4	13.7	2410.0	584.0	1.69	0.734
n = 28	76.2	19.0	2850.0	820.0	2.0	0.97
n = 30	76.1	11.1	3060.0	676.0	2.06	0.691
n = 32	75.0	8.95	3330.0	645.0	2.27	0.584
n = 34	83.5	9.44	4050.0	783.0	2.53	0.812

	Percentage Correct		
Size	LV	Continuation	Greedy
n = 10	0.922	0.996	0.978
n = 12	0.92	0.98	0.962
n = 14	0.918	0.97	0.944
n = 16	0.89	0.954	0.93
n = 18	0.882	0.936	0.928
n = 20	0.874	0.948	0.936
n = 22	0.85	0.922	0.902
n = 24	0.854	0.928	0.91
n = 26	0.84	0.891	0.868
n = 28	0.83	0.898	0.848
n = 30	0.766	0.877	0.844
n = 32	0.758	0.843	0.84
n = 34	0.736	0.86	0.768

	Worstcase Performance		
Size	LV	Continuation	Greedy
n = 10	0.6	0.8	0.8
n = 12	0.75	0.8333	0.8
n = 14	0.7143	0.8333	0.8333
n = 16	0.75	0.8571	0.75
n = 18	0.75	0.8571	0.7778
n = 20	0.7778	0.875	0.8
n = 22	0.8	0.9	0.8182
n = 24	0.8333	0.8333	0.7273
n = 26	0.8462	0.8462	0.8333
n = 28	0.8571	0.8571	0.8333
n = 30	0.8667	0.8571	0.7857
n = 32	0.8571	0.8667	0.8667
n = 34	0.8571	0.875	0.8667

A.2 Geometric graphs data

Average Performance						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 10	0.9349	0.1266	0.9966	0.02861	0.9968	0.02893
n = 12	0.9294	0.1196	0.9908	0.04304	0.9978	0.02005
n = 14	0.9362	0.1089	0.9899	0.04309	0.9962	0.02662
n = 16	0.9135	0.1177	0.9918	0.03588	0.9953	0.02929
n = 18	0.9079	0.1065	0.9894	0.03988	0.9943	0.02901
n = 20	0.9177	0.1077	0.984	0.04632	0.9944	0.0274
n = 22	0.9192	0.1026	0.9854	0.04183	0.9925	0.03115
n = 24	0.9024	0.102	0.98	0.04801	0.9937	0.0278
n = 26	0.9032	0.1051	0.9768	0.04937	0.9898	0.03367
n = 28	0.9008	0.0944	0.9759	0.0479	0.989	0.03373
n = 30	0.9196	0.09103	0.9755	0.04796	0.9881	0.03554
n = 32	0.8959	0.09803	0.975	0.04671	0.9908	0.03047
n = 34	0.8997	0.09353	0.9698	0.04939	0.9893	0.03205

Average Running Time (10^{-3} s)						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 10	68.3	25.3	774.0	306.0	0.49	0.515
n = 12	62.1	15.7	873.0	259.0	0.557	0.608
n = 14	62.7	17.1	1080.0	317.0	0.654	0.659
n = 16	58.8	8.27	1190.0	291.0	0.704	0.478
n = 18	72.7	31.6	1760.0	725.0	1.09	0.95
n = 20	75.2	28.1	2090.0	856.0	1.17	0.816
n = 22	66.8	13.6	2180.0	578.0	1.11	0.504
n = 24	67.2	17.0	2500.0	687.0	1.33	0.872
n = 26	74.4	24.2	3050.0	748.0	1.52	0.883
n = 28	77.0	22.0	3570.0	1050.0	1.66	0.809
n = 30	81.4	23.7	4150.0	1130.0	1.77	0.784
n = 32	75.1	24.5	4420.0	1070.0	1.94	1.53
n = 34	88.3	29.9	7400.0	40400.0	2.13	0.766

	Percentage Correct		
Size	LV	Continuation	Greedy
n = 10	0.77	0.986	0.988
n = 12	0.712	0.956	0.988
n = 14	0.714	0.948	0.98
n = 16	0.598	0.95	0.974
n = 18	0.518	0.932	0.962
n = 20	0.562	0.891	0.96
n = 22	0.552	0.891	0.944
n = 24	0.43	0.846	0.95
n = 26	0.438	0.815	0.916
n = 28	0.38	0.794	0.904
n = 30	0.476	0.786	0.896
n = 32	0.36	0.772	0.914
n = 34	0.342	0.719	0.898

	Worstcase Performance		
Size	LV	Continuation	Greedy
n = 10	0.3333	0.75	0.6667
n = 12	0.3333	0.75	0.8
n = 14	0.4	0.75	0.75
n = 16	0.4	0.8	0.7143
n = 18	0.5	0.75	0.8333
n = 20	0.4	0.8	0.8333
n = 22	0.5714	0.8333	0.8333
n = 24	0.4286	0.7143	0.8571
n = 26	0.5	0.75	0.8571
n = 28	0.625	0.7778	0.875
n = 30	0.5556	0.8	0.7778
n = 32	0.5556	0.7778	0.8
n = 34	0.5	0.7778	0.8

A.3 Bipartite graphs data

Average Performance						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 20	0.9875	0.02976	0.9983	0.01167	0.985	0.05449
n = 30	0.9878	0.02301	1.0	0.0	0.9806	0.06353
n = 40	0.9883	0.02211	1.0	0.0	0.9642	0.08123
n = 50	0.9887	0.02222	1.0	0.0	0.9723	0.07289
n = 60	0.9939	0.01448	1.0	0.0	0.9714	0.06217
n = 70	0.9867	0.01823	1.0	0.0	0.9726	0.06846
n = 80	0.99	0.01487	1.0	0.0	0.9679	0.06077
n = 90	0.9924	0.01286	0.9998	0.001843	0.9661	0.05708
n = 100	0.9905	0.0162	1.0	0.0	0.9697	0.05428

Average Running Time (10^{-3} s)						
	LV		Continuation		Greedy	
Size	Mean	Std	Mean	Std	Mean	Std
n = 20	88.0	41.8	1430.0	521.0	1.54	0.684
n = 30	78.9	17.3	2150.0	372.0	2.45	1.34
n = 40	125.0	33.5	4670.0	807.0	4.69	1.87
n = 50	162.0	24.6	7980.0	901.0	6.85	2.47
n = 60	187.0	34.2	11800.0	1250.0	9.17	3.11
n = 70	231.0	33.8	18000.0	1780.0	12.1	4.24
n = 80	272.0	37.8	25800.0	2230.0	14.2	4.44
n = 90	357.0	48.9	37500.0	3280.0	18.3	4.57
n = 100	395.0	77.1	52000.0	3690.0	22.2	6.08

	Percentage Correct		
Size	LV	Continuation	Greedy
n = 20	0.85	0.98	0.92
n = 30	0.78	1.0	0.89
n = 40	0.76	1.0	0.74
n = 50	0.75	1.0	0.79
n = 60	0.82	1.0	0.69
n = 70	0.58	1.0	0.74
n = 80	0.63	1.0	0.6
n = 90	0.69	0.99	0.53
n = 100	0.63	1.0	0.49

	Worstcase Performance		
Size	LV	Continuation	Greedy
n = 20	0.9167	0.9167	0.75
n = 30	0.9444	1.0	0.6667
n = 40	0.9167	1.0	0.6667
n = 50	0.9	1.0	0.6667
n = 60	0.9167	1.0	0.6667
n = 70	0.9286	1.0	0.6667
n = 80	0.9375	1.0	0.6667
n = 90	0.9444	0.9815	0.6852
n = 100	0.9167	1.0	0.6667

A.4 Algorithm comparison

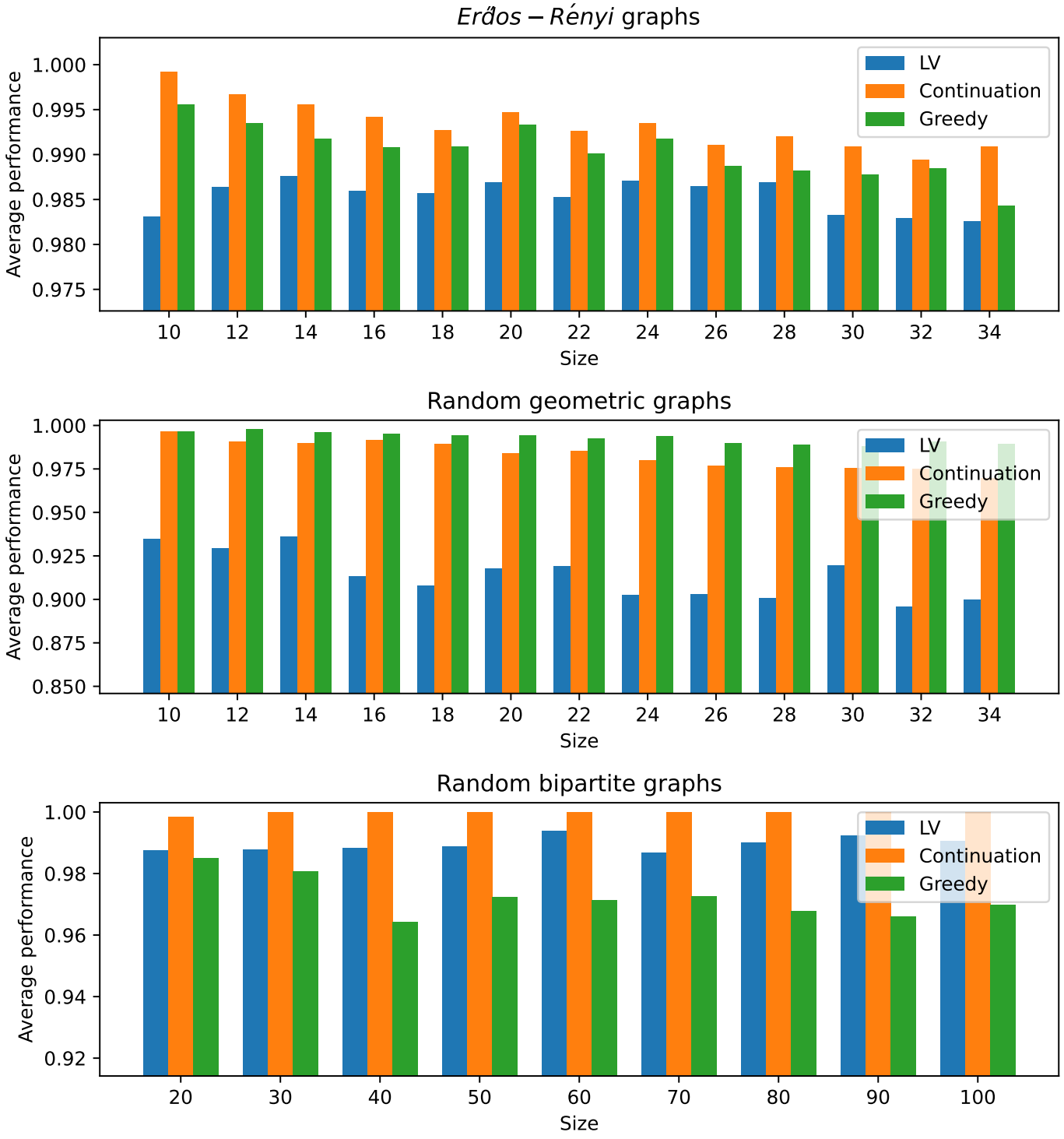


Figure A.4.1: Average performance of the LV algorithm, the Continuation algorithm and the Greedy algorithm.

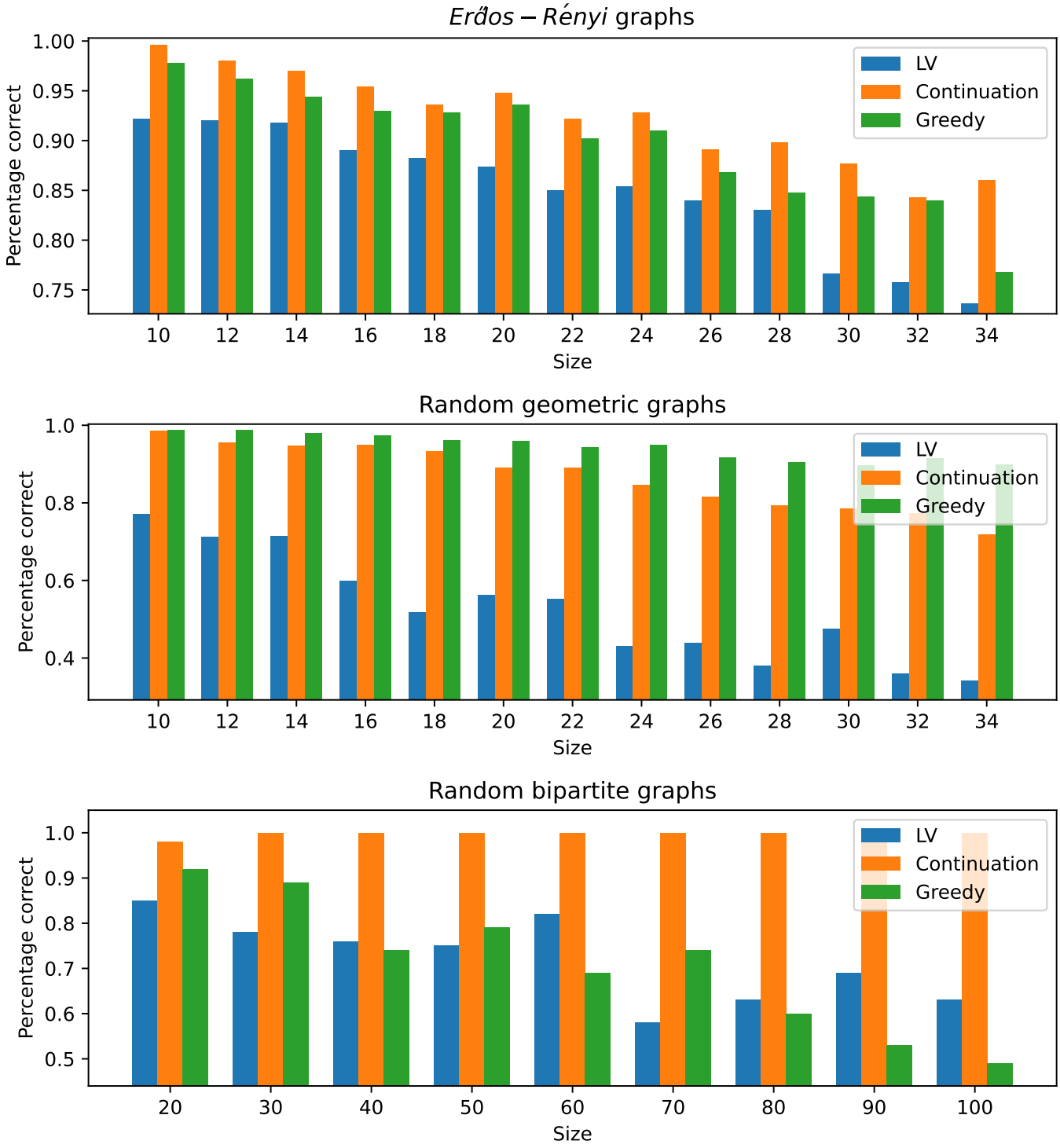


Figure A.4.2: Percentage correct of the LV algorithm, the Continuation algorithm and the Greedy algorithm.

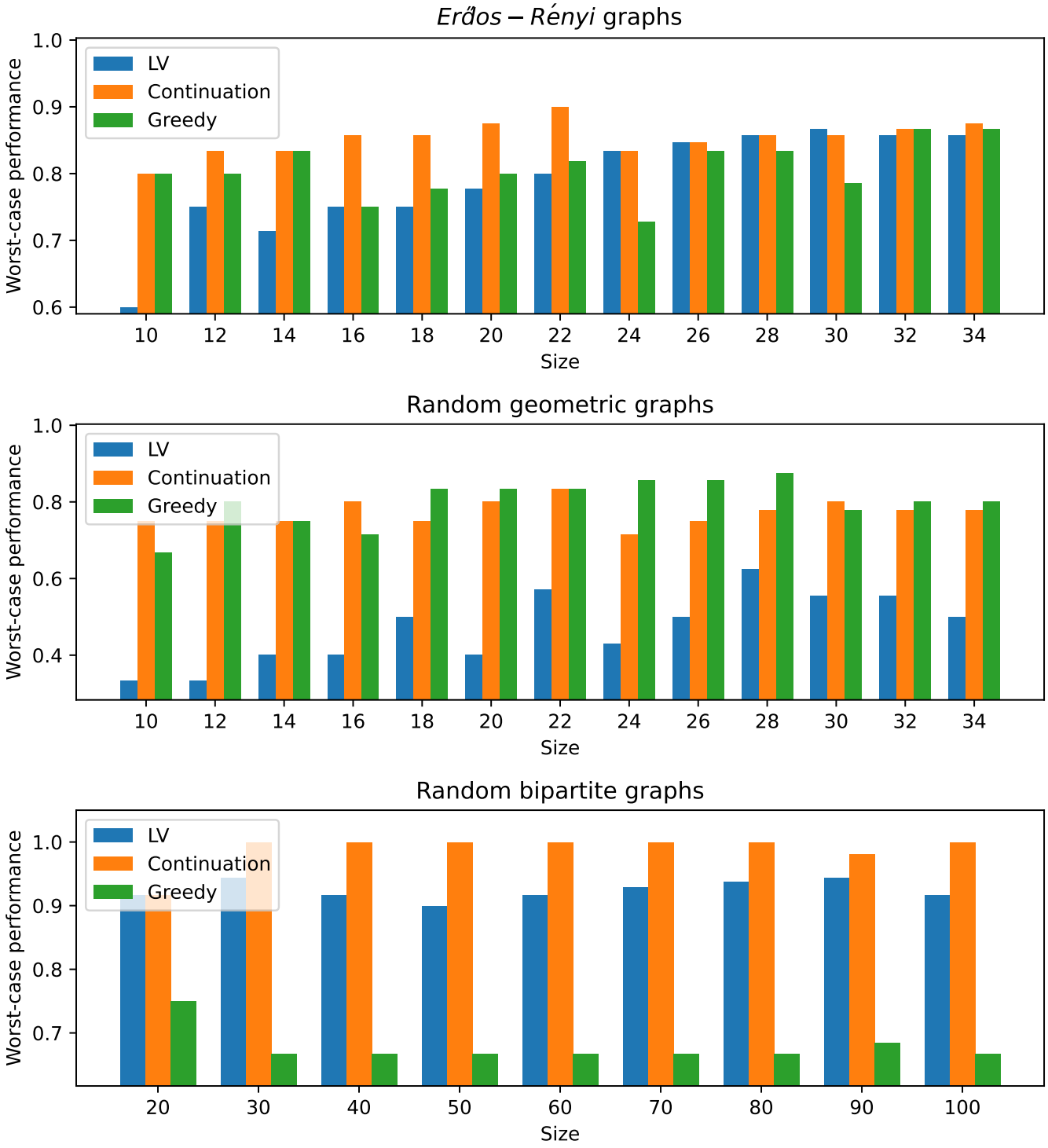


Figure A.4.3: Worst-case performance of the LV algorithm, the Continuation algorithm and the Greedy algorithm.

References

[1] S. Ağrali, Z. Caner Taşkin, and A. Tamer Ünal. "Employee scheduling in service industries with flexible employee availability and demand". In: *Omega* 66 (2017), pp. 159–169. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2016.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0305048316000475.

[2] K. Ameenal Bibi, A. Lakshmi, and R. Jothilakshmi. "Applications of Distance - 2 Dominating Sets of Graph in Networks". In: *Advances in Computational Sciences and Technology* 10.9 (2017), pp. 2801–2810.

[3] P. Ashwin and M. Timme. "When instability makes sense". In: *Nature* 436 (July 2005), pp. 36–37.

[4] T. Bäck and S. Khuri. "An evolutionary heuristic for the maximum independent set problem". 1994.

[5] B. S. Baker. "Approximation Algorithms for NP-Complete Problems on Planar Graphs". In: *Journal of the Association for Computing Machinery* 41.1 (Jan. 1994), pp. 153–180.

[6] S. Balaji, V. Swaminathan, and K. Kannan. "A simple algorithm to optimize maximum independent set". In: *Advanced Modeling and Optimization* 12.1 (2010), pp. 107–118.

[7] J. C. Ballard-Myer. "Deterministic Greedy Algorithm for Maximum Independent Set Problem in Graph Theory". Dec. 2019.

[8] I. M. Bomze. "Lotka-Volterra Equation and Replicator Dynamics: A Two-Dimensional Classification". In: *Biological Cybernetics* 48 (1983), pp. 201–211.

[9] I. M. Bomze et al. "The Maximum Clique Problem". In: *Handbook of Combinatorial Optimization*. Ed. by D. Ding-Zhu and P. M. Pardalos. Springer, 1999, pp. 1–74. DOI: 10.1007/978-1-4757-3023-4_1. URL: https://doi.org/10.1007/978-1-4757-3023-4%5C_1.

[10] A. Brændeland. *A family of greedy algorithms for finding maximum independent sets*. 2015. arXiv: 1505.00752 [cs.DS].

[11] G. Bunin. "Ecological communities with Lotka-Volterra dynamics". In: *Physical Review E* 95.4 (2017), p. 042414.

[12] S. Butenko. *Maximum independent set and related problems, with applications*. University of Florida, 2003.

[13] B. Chamaret et al. "Radio network optimization with maximum independent set search". In: *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*. Vol. 2. IEEE. 1997, pp. 770–774.

[14] S. Cook. "The importance of the P versus NP question". In: *Journal of the ACM (JACM)* 50.1 (2003), pp. 27–29.

[15] W. J. Cook et al. "Combinatorial optimization". In: *Oberwolfach Reports* 5.4 (2009), pp. 2875–2942.

[16] P. Erdos, A. Rényi, et al. "On the evolution of random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[17] M. R. Garey and D. S. Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[18] S. A. Gershgorin. "Uber die abgrenzung der eigenwerte einer matrix". In: . 6 (1931), pp. 749–754.

[19] G. T. Gilbert. "Positive definite matrices and Sylvester's criterion". In: *The American Mathematical Monthly* 98.1 (1991), pp. 44–46.

[20] B. S. Goh. "Global stability in many-species systems". In: *The American Naturalist* 111.977 (1977), pp. 135–143.

[21] A. V. Goldberg, É. Tardos, and R. Tarjan. *Network flow algorithm*. Tech. rep. Cornell University Operations Research and Industrial Engineering, 1989.

[22] J. L. Gross, J. Yellen, and M. Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[23] M. Gupta and S. Khan. "Simple dynamic algorithms for maximal independent set and other problems". In: *arXiv preprint arXiv:1804.01823* (2018).

[24] S. M. Hedetniemi et al. "Self-stabilizing algorithms for minimal dominating sets and maximal independent sets". In: *Computers & Mathematics with Applications* 46.5-6 (2003), pp. 805–811.

[25] M. W. Hirsch. "Systems of differential equations which are competitive or cooperative: III. Competing species". In: *Nonlinearity* 1.1 (1988), p. 51.

[26] K. Huseyin and R. Plaut. "Application of the Rayleigh quotient to eigenvalue problems of pseudo-conservative systems". In: *Journal of Sound and Vibration* 33.2 (1974), pp. 201–210.

[27] R. Karp. "Reducibility Among Combinatorial Problems". In: vol. 40. Jan. 1972, pp. 85–103. ISBN: 978-3-540-68274-5. DOI: 10.1007/978-3-540-68279-0_8.

[28] E. Kaszkurewicz and A. Bhaya. *Matrix diagonal stability in systems and computation*. Springer Science & Business Media, 2012.

[29] J. H. Kim and V. H. Vu. "Generating random regular graphs". In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, pp. 213–222.

[30] L. Lovász and J. Pelikán. "On the eigenvalues of trees". In: *Periodica Mathematica Hungarica* 3.1-2 (1973), pp. 175–182.

[31] R. Mac Arthur. "Species Packing, And What Interspecies Competition Minimizes". In: *Proceedings of the National Academy of Sciences of the United States of America* 64.4 (1969). https://doi.org/10.1073/pnas.64.4.1369, pp. 1369–1371.
[32] G. Mao. *Connectivity of communication networks*. Springer, 2017.
[33] B. D. McKay. "The expected eigenvalue distribution of a large regular graph". In: *Linear Algebra and its Applications* 40 (1981), pp. 203–216.
[34] R. Menezes dos Santos. "Exploring ecological interactions using the generalized Lotka-Volterra model - Coexistence and resilience of populations". Master thesis. Federal University of Bahia, Jan. 2021.
[35] V. Nikiforov. "The influence of Miroslav Fiedler on spectral graph theory". In: *Linear Algebra and Its Applications* 439.4 (2013), pp. 818–821.
[36] M. Pelillo. "Heuristics for maximum clique and independent set". In: (1999).
[37] J. Quandt. "On the Hartman-Grobman theorem for maps". In: *Journal of differential equations* 64.2 (1986), pp. 154–164.
[38] J. M. Robson. "Algorithms for maximum independent sets". In: *Journal of Algorithms* 7.3 (1986), pp. 425–440.
[39] A. Sharp. *Proving a Problem is NP-complete*. Accessed: 9-6-2022.
[40] I. Stojmenovic, M. Seddigh, and J. Zunic. "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks". In: *IEEE Transactions on parallel and distributed systems* 13.1 (2002), pp. 14–25.
[41] S. H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.
[42] M. Uetz. *Discrete Optimization 2020 - Lecture 9: P, NP, co-NP, and Problem Reductions*. Mastermath Discrete Optimization Course. 2020.
[43] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018.
[44] T. Vinh-Thong, A. Elmoataz, and O. Lézoray. "Nonlocal PDEs-based morphology on weighted graphs for image and data processing". In: *IEEE transactions on Image Processing* 20.6 (2010), pp. 1504–1516.
[45] A. Wigderson. "P, NP and mathematics–a computational complexity perspective". In: *Proceedings of the ICM*. Vol. 6. 2006, pp. 665–712.
[46] R. J. Wilson. "1.3 History of Graph Theory". In: *Handbook of Graph Theory* (2003), p. 29.
[47] M. Xiao and H. Nagamochi. "Exact algorithms for maximum independent set". In: *Information and Computation* 255 (2017), pp. 126–146.