Pretopology, a mathematical tool for structuring complex systems : methods, algorithms and applications

Julio Laborde

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'École Pratique des Hautes Études

Pretopology, a mathematical tool for structuring complex systems: methods, algorithms and applications

École doctorale de l'EPHE - ED 472

Spécialité INFORMATIQUE, STATISTIQUES ET COGNITION

COMPOSITION DU JURY :

Mme. Nahid Emad
Université de Versailles
Saint-Quentin-en-Yvelines
Rapporteur

M. Mhand Hifi
Université de Picardie Jules Verne
Rapporteur

Mme. Isis Truck
Université Paris 8
Membre du jury

Mme. Nadia Kabachi
Université Claude Bernard Lyon 1
Membre du jury

M. François Dubois
Université Paris Sud, Orsay
Président du jury

M. Marc Bui
École Pratique des Hautes Études
Directeur de thèse

Soutenue par **Julio Laborde**
le 02 avril 2019

Dirigée par **Marc BUI**

École Pratique des Hautes Études

PSL RESEARCH UNIVERSITY PARIS

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When we study a complex system it is natural to be interested in the organization or structure of its elements, as opposed to just caring about the individual characteristics of the components. The traditional way to deal with this issue has been through a characterization of the structure by the use of graphs (or networks), where nodes represent the components of the system, and an edge exists between two of them if there is a relation connecting them.

Network theory has proved to be extremely productive dealing with a huge number of problems from a great number of fields. The theory is nevertheless only appropriate for the description of systems with binary underlying relations between the components. Well, manifold situations have more complex relations among the elements.

Multilayer networks could be used for the study of many types of binary links simultaneously, and when the connections are not binary but multiple one could use hypergraphs. None of these frameworks seems to effectively treat the case where connections exist between an element and a group. We propose here the use of pretopology to fill this gap.

A pretopological space, in its most general form, is defined only by a pseudoclosure function that associates any set to an equal or bigger set. Three other axioms can be added in order to get less general spaces with more powerful theorems. By combining some of these axioms one can get the definition of a graph or that of a topological space. Pretopology is then a generalization of those two theories.

For some spaces we can get an equivalent definition by associating a family of sets (*i.e.* the basis of a prefilter of neighborhoods) to each element, and joining to the pseudoclosure of a set all those elements that intersect the set with every neighborhood. This other way of conceiving the problem, where instead of interrogate a set about its pseudoclosure, we ask every external element whether it belongs to it or not, has proven to be very productive and it is the one that is usually employed in practice.

Many generalizations of this last notion exist where rather than asking for every neighborhood to intersect the set, we state more complex rules to decide if an element belongs to the pseudoclosure. This has allowed to have more flexibility when defining a pretopological space in exchange of the mathematical equivalence between both definitions.

In its functional version a pretopology presents itself very naturally as a framework to study dynamic phenomena, such as the diffusion or expansion of a part of a system. The presentation in terms of neighborhoods, on the other hand, invites to model systems where relations exist between an element and a set, either because the relation can be naturally expressed that way, or because it emerges from a combination of multiple binary relations.

It is interesting to realize how in the same way a directed graph can be used to study the influence of the components of a system -which is something static-, and problems of flux optimization -which is a dynamic phenomena-, also pretopology can use the same mathematical object to study the fixed structure and the dynamics of a system depending on the interpretation we give to the components.

Although pretopology has already been successfully used as a modeling tool on a number of different real problems, its popularity is still quite inferior than that of the other theories previously mentioned. We believe this is partly due to the complexity of its definition in terms of sets. Indeed, in its functional version, only the definition of the space would need to list the image under the pseudoclosure function for every possible subset of the system. Now, that list becomes intractable extremely fast.

In order to overcome that problem, researchers have generalized more and more the definition in terms of neighborhoods. None of those generalizations though, includes all of the others. We found ourselves then with a great number of conceptually similar studies (they all use the language of pretopology), but with little in common from a practical point of view: each work represents the space in the computer in a different fashion -usually without giving details-, and implements the traditional metrics in its own way. The inconvenience of this is obvious: not only there is a lot of redundant work, since we have to remake what has already been done many times, but also it becomes very difficult to build on the work of others.

The goals of this thesis are three: the first is to give an answer to the need of homogenization that was just evoked; the second is to improve the existent results from an algorithmic point of view; and the last one is to use the theory in an original form to solve some real problems.

The PhD is structured in the following way: we begin by describing the landscape of structural frameworks and some of their applications. This should convince us of the utility and importance of choosing the right framework for the study of a problem.

We then present our contributions, starting with our formalization of a pretopological space in terms of a simple group of rules over a set of networks. We show the generality of this procedure by presenting a great number of previous works formalized in this manner.

The rest of the contributions section introduce or discuss some metrics and algorithms over a pretopological space. Some of the algorithms are improved, and the use of our framework shows its first utility by allowing us to study more precisely the complexity of the calculations.

The last part is dedicated to applications. The first of them is a Python library where all of the previously reviewed algorithms are implemented. After presenting in a general manner the faculties of the library, we use it to study some diffusion phenomena with the help of some agent based models.

A final application concerns clusterization. We use some of the notions introduced here for the first time to develop a clusterization algorithm, that not only performs on a par with the state of the art on some artificial geometrical data, but also has the advantage of being immediately generalizable to non-metrical spaces.

The document ends discussing the results obtained, as well as a great number of research subjects that follow naturally from the framework and results presented here.

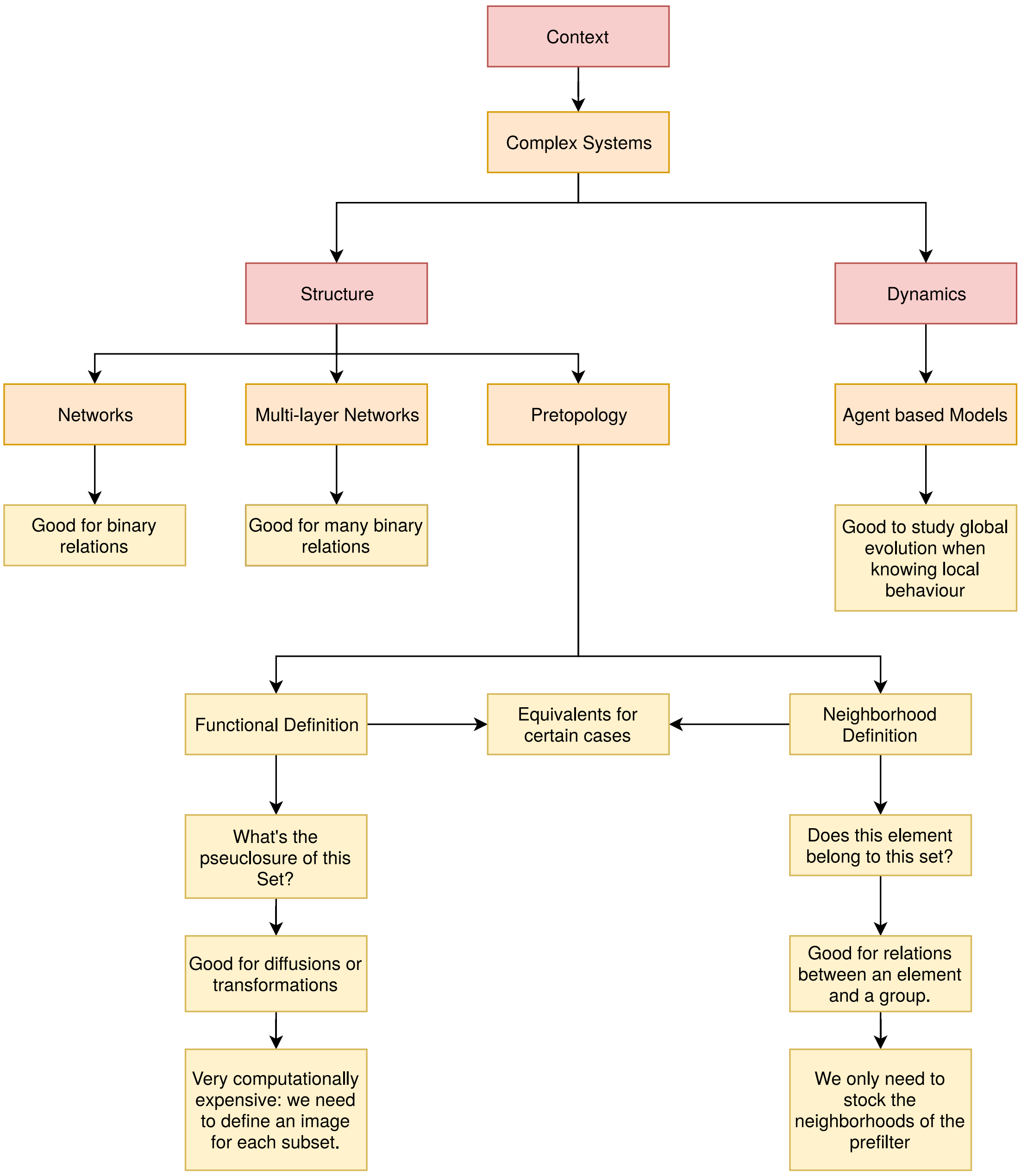
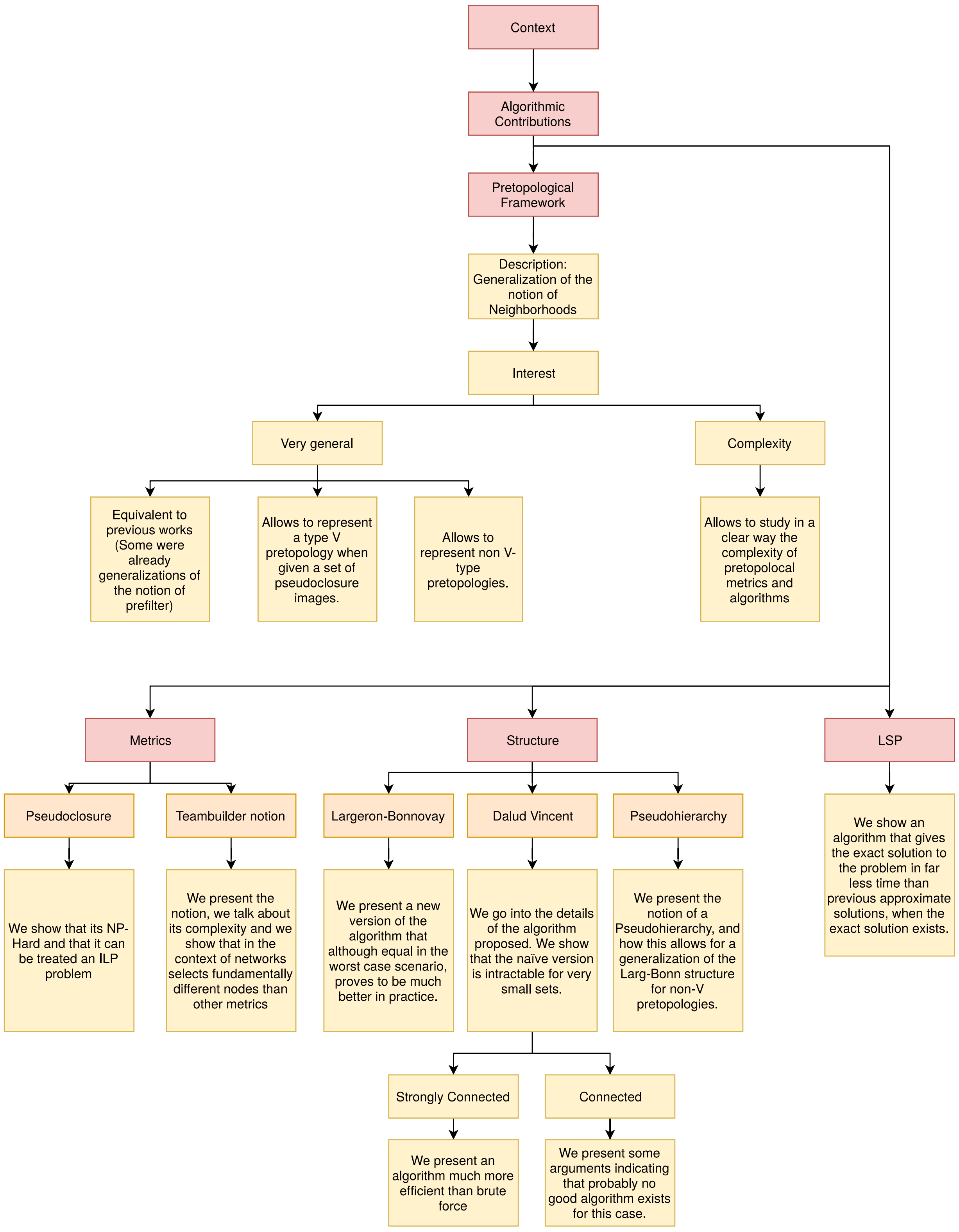

Context
Complex Systems
Structure
Dynamics
Networks
Multi-layer Networks
Pretopology
Agent based Models
Good for binary relations
Good for many binary relations
Good to study global evolution when knowing local behaviour
Functional Definition
Equivalents for certain cases
Neighborhood Definition
What's the pseuclosure of this Set?
Does this element belong to this set?
Good for diffusions or transformations
Good for relations between an element and a group.
Very computationally expensive: we need to define an image for each subset.
We only need to stock the neighborhoods of the prefilter

Figure 1.1: Schematic representation of the context



Context
Algorithmic Contributions
Pretopological Framework
Description: Generalization of the notion of Neighborhoods
Interest
Very general
Complexity
Equivalent to previous works (Some were already generalizations of the notion of prefilter)
Allows to represent a type V pretopology when given a set of pseudoclosure images.
Allows to represent non V-type pretopologies.
Allows to study in a clear way the complexity of pretopolocal metrics and algorithms
Metrics
Structure
LSP
Pseudoclosure
Teambuilder notion
Largeron-Bonnovay
Dalud Vincent
Pseudohierarchy
We show an algorithm that gives the exact solution to the problem in far less time than previous approximate solutions, when the exact solution exists.
We show that its NP-Hard and that it can be treated an ILP problem
We present the notion, we talk about its complexity and we show that in the context of networks selects fundamentally different nodes than other metrics
We present a new version of the algorithm that although equal in the worst case scenario, proves to be much better in practice.
We go into the details of the algorithm proposed. We show that the naïve version is intractable for very small sets.
We present the notion of a Pseudohierarchy, and how this allows for a generalization of the Larg-Bonn structure for non-V pretopologies.
Strongly Connected
Connected
We present an algorithm much more efficient than brute force
We present some arguments indicating that probably no good algorithm exists for this case.

Figure 1.2: Schematic representation of the algorithmic contributions

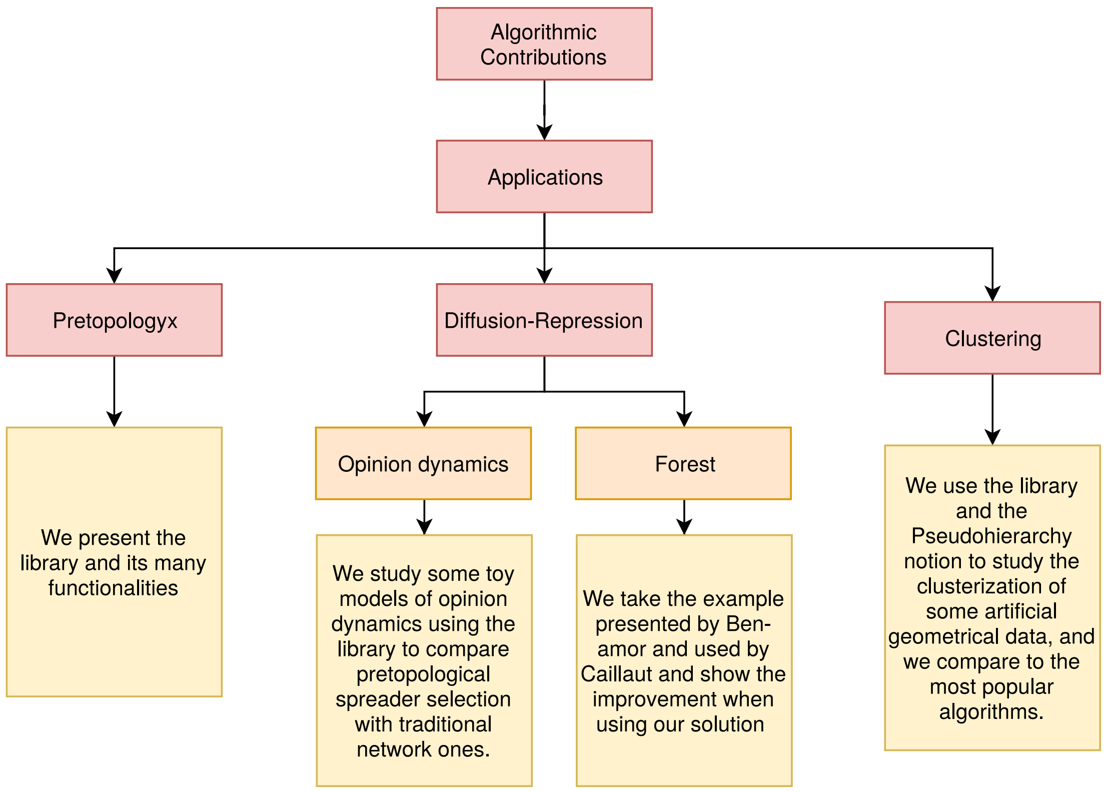

Algorithmic Contributions
Applications
Pretopologyx
Diffusion-Repression
Clustering
We present the library and its many functionalities
Opinion dynamics
Forest
We study some toy models of opinion dynamics using the library to compare pretopological spreader selection with traditional network ones.
We take the example presented by Ben-amor and used by Caillaut and show the improvement when using our solution
We use the library and the Pseudohierarchy notion to study the clusterization of some artificial geometrical data, and we compare to the most popular algorithms.

Figure 1.3: Schematic representation of the applications

Part I

Context

Chapter 2

Structural Frameworks

This chapter will present the context necessary to understand the main premise of the thesis, that pretopology can be a useful structural framework, *i.e.* a mathematical object used to characterize the relations of the parts of a system.

The context is divided in two parts. First, we give a brief recount of graph theory and some related theories. This has three objectives:

- To understand the role and the benefits of choosing a good framework to study the relations inside a system.
- To understand the limitations of these theories, and see what could be done to overcome them.
- To have in mind these more popular theories, so they can be used as analogies or points of reference when trying to understand the concepts of pretopology.

The second part presents the definition of a pretopological space and some of the concepts of the theory. We also mention a few applications to render the definitions more concrete.

2.1 Graphs

A graph $G(V, E)$ is a mathematical object composed of a set V of elements called nodes, and a set $E \subset V \times V$ of unordered pairs of nodes called edges [21]. We say that an edge connects a pair of nodes. Graphs can be easily drawn by associating nodes to points or circles, and lines between them when there is an edge connecting them. A draw of a graph can be seen in figure 2.1.

A graph is said to be **directed** if the set of its edges is composed of ordered pairs. We say then that an edge x_0x_1 goes from x_0 to x_1 and we draw it with an arrow pointing from x_0 to x_1.

A graph is said to be **weighted** if each edge has a number associated to it.

Definition 2.1. *A path is a graph* $\mathbf{P}$ *of the form:*

$$V(\mathbf{P}) = \{x_0, x_1, \ldots, x_l\}, \quad E(P) = V(\mathbf{P}) = \{x_0x_1, x_1x_2, \ldots, x_{l-1}x_l\}$$

This path $\mathbf{P}$ is usually denoted by $x_0x_1 \ldots x_l$ and its length is the cardinality of the set of edges. Simply put, a path is a tuple of edges, where each edge -except the first and the last one- has one vertex in common with the previous edge, and the other in common with the following edge.

Ideas related to graph theory can be traced back as far as the eighteenth century, with the famous *seven bridges problem of Königsberg*. The city of Königsberg had the same configuration than the center of Paris, with two river banks and two small islands, and had seven bridges connecting the four different pieces of land. There was the question of the possibility to make a walk that would cross each bridge exactly once. Leonard Euler solved the problem negatively by associating a graph to the city: nodes were the pieces of land, edges were the bridges.

Although the idea of a graph has almost three centuries, it was only in 1936 that Dénes Kőnig published the first book on what is now called graph theory. On the other hand, also in the thirties, Moreno was introducing the concept of *sociogram*, which was a diagram of points and lines representing relations among people, a precursor to the graph representation for a system.

Nowadays, when a graph is applied to describe a real system it is usually called a network [131, 12]. In these cases we associate the components with nodes, and edges with connections or relations between the components. Although the difference between a graph and a network may play a role sometimes, as when studying the complexity of graph algorithms that work well on real cases but have bad worst case performance [48], during the thesis we will use both concepts more or less indistinctly.

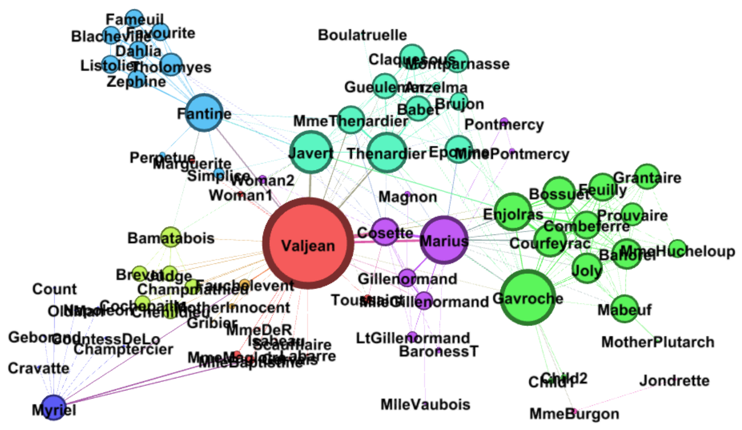


Figure 2.1: Example of a Graph

2.1.1 Metrics

When we associate a graph to a system, when can start studying the metrics (or measures) of the graph to see what they can teach us about the system. A metric is a calculation defined on the graph in order to quantitatively differentiate nodes according to their structural position, or to differentiate among diverse graphs. Most of these different calculations are defined so they can capture in an unambiguous fashion some natural concepts used to qualify the members of a system.

Centrality [24, 25] is one of the most important and widely used conceptual tools for analyzing networks. Although many different definitions of what it means to be central exist, they all share the intention of assigning a bigger centrality score to the more important nodes. It is off course the notion of importance that changes according to the system or phenomenon one is trying to model.

In 1979, Freeman [62] put some order in the multitude of centrality measures proposed, categorizing some of them into three basic classes: degree, closeness and betweenness. Although these too are notions that have sometimes more than one definition [105], the most canonical definitions are the following:

- The **degree** of a node is the number of neighbors the node has in the graph, that is, the number of nodes that are connected to the node by an edge. In a Social network, a node with a high degree represents an individual that knows a lot of people, and for that same reason he/she is likely to be an important person.

- The **betweenness centrality** is defined as follows:

$$b(x) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

  Where σ_{st} is the number of shortest paths between s and t, and $\sigma_{st}(v)$ is the number of those paths that passes through v. In other words, the betweenness centrality of a node n_0 is computed by taking the shortest paths between every couple of nodes in the graph, and calculating the fraction of them that include n_0. If the fraction is high it would mean that when two nodes want to communicate in an optimal way (through their shortest path), then there is a good chance that they will pass through this node. A node with these characteristics could have capital importance if we wanted to hinder the communication inside of a network.

- The **closeness centrality** is defined as follows:

$$C(x) = \frac{1}{\sum_y d(y, x)}$$

  It is the inverse of the sum of the distances to every other node. So nodes will be more central, the shorter the distance to every other node in the network. Importance here can translate the capacity to communicate an information to every other node in the fastest possible manner.

The metrics we have mentioned so far concern particular nodes. Other metrics try to capture properties of the network as a whole. One such property is the clustering coefficient of the network $C = \frac{3 \times Number\ of\ triangles}{Number\ of\ connected\ triplets}$. This tries to measure how likely is for two neighbors of a node to be neighbors among them. This metric is usually important because it allows to uncover a phenomenon that we find very frequently on social networks; indeed, it is a lot more likely for two friends of someone to be acquainted among them, than for two randomly selected people.

Figure 2.2 shows a graphical representation of some of the metrics presented.

2.1.2 Spectral Graph Theory

Graphs can also be expressed in terms of matrices, and this other representation also allows to get important insights into the structure of the system we are modeling. The study of a graph using matrices is called spectral graph theory.

The most fundamental matrix associated to a graph is the adjacency matrix of the graph. This is a $|V| \times |V|$ square matrix $\mathbf{A}$, such that $\mathbf{A}_{ij} = 1$ if there is an edge going from node i to node j.

As a simple example of the utility of the adjacency matrix, we can mention how A^n gives us the number of paths of length n between any pair of elements.

One of the most interesting applications of the adjacency matrix is in the context of the centrality of a node. Let's suppose that every node has a value that quantifies how important it is, and let's suppose in addition that the importance of a node is

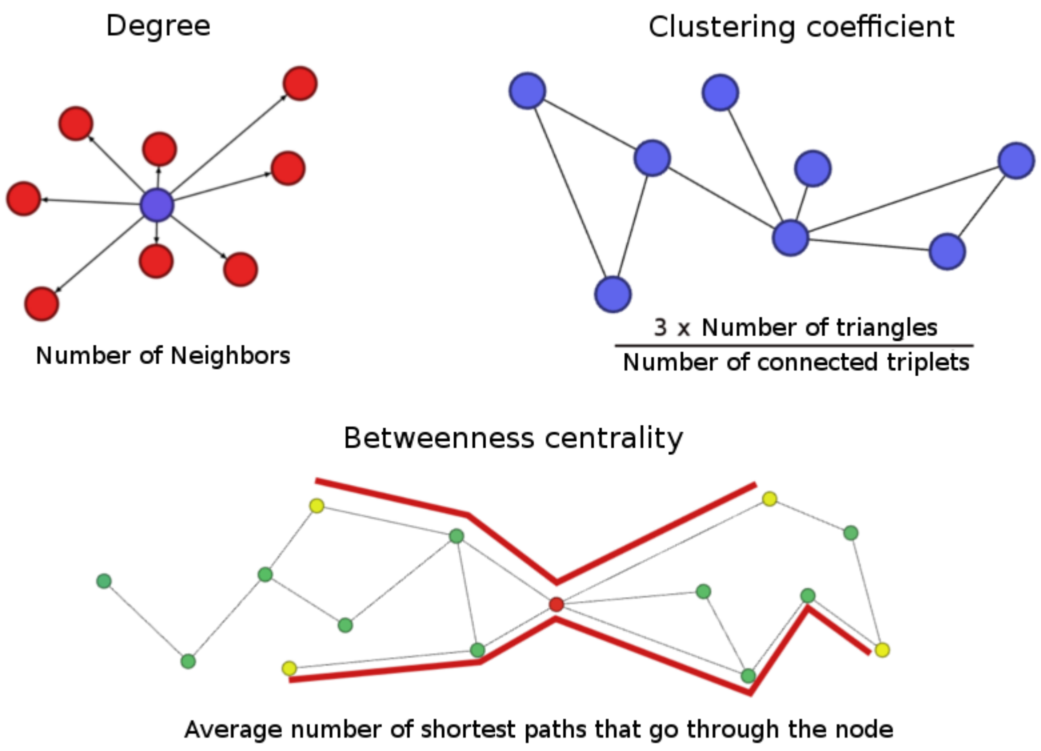


Figure 2.2: Graph Metrics

proportional by a factor of λ to how important are the nodes connected to it. If we put the importance of the nodes in a vector **x**, then **x** would comply with the following equation:

$$Ax = \lambda x$$

Now, solving this equation boils down to finding an eigenvalue and an eigenvector for the linear transformation defined by the adjacency matrix. This is called eigenvector centrality. Variations of this idea can be found in the Katz centrality, and the very popular pagerank [20], that is at the base of google.

2.1.3 Network Models

So far, the way of extracting information from a system by the use of networks, has been by identifying a graph with the system at a point in time, and see what kind of static properties that graph has. There are other ways in which network theory has helped understanding some complex systems, where instead of having a single network to which we apply a metric, we have a model explaining the origin of the network. These are called dynamic network models [3, 65, 4], since they are concerned with the mechanisms governing changes in the network over time.

The most paradigmatic works in that field come from the late nineties and the beginning of this century, where physicists started applying the same kind of thinking they were using in statistical mechanics to the study of real networks. This approach was not that different from some work on random graphs from the sixties (c.r. see bellow), where parameters of a model defining a graph were tuned to study some emergent properties, but the motivations where different. The interest being applicative in the former, and mathematical in the later.

The reasons for this change of interest are certainly related to the development of computers during those decades. Computers and their ubiquitous use had changed the possibility of studying networks in three very concrete ways: they had created new complex systems (the WWW and the Internet being two emblematic examples), they were allowing to gather more and more data about those and other systems; and they were allowing to make huge calculations numerically.

We will now briefly examine three of those dynamical models, and see how each of them allowed to better understand the origin of some empirically observed properties of real networks.

2.1.3.1 Erdős-Rényi

This model was presented in 1959 [55], and although its motivation was mostly mathematical, it was this model that guided the thinking about complex networks for decades since its introduction [4].

The model starts with N nodes and connects every pair of nodes with probability p, resulting in a graph with approximately $p\frac{N(N-1)}{2}$ nodes distributed randomly. The distribution of the degrees over the whole graph, for N sufficiently large, approximates a Poisson distribution.

The authors studied then a series of properties of the emergent graphs for different values of p. Probably the most interesting of those results concerns the existence of a threshold of connectivity. Indeed, the authors had shown that:

- If $p < \frac{(1-\varepsilon)\ln n}{n}$, the graph will be almost surely disconnected.
- If $p > \frac{(1-\varepsilon)\ln n}{n}$, the graph will be almost surely connected.

Proving this way that $\frac{(1-\varepsilon)\ln n}{n}$ is a threshold for the connectedness of the graph.

2.1.3.2 Watts-Strowgatz

In 1998 Watts and Strowgatz presented a model that allowed to study in a formal way the so called *small world* phenomenon [133].

Already in 1929 the Hungarian writer Frigyes Karinthy [103] had published a short story arguing how closely we are all connected to each other in the world. The story presents two characters discussing, one of them telling the other that a person was probably not more than five or six steps away from any other, if they followed the path of social acquaintances. After discussing two examples, the other person seems more or less convinced about the truth of the statement.

A couple of decades later, in 1967 [126], Milgram tried to frame this same idea into a more scientific setting. He set up an experiment where a letter with a name and an address was given to a few hundreds of people randomly selected in some state. The person was instructed to send this letter to someone he knew, and he thought to have more chances of knowing the individual whose name was on the letter. By collecting the letters arriving to the destination, and studying the intermediary senders, Milgram was able to study the average degree of separation between people. Although the experiment was far from perfect, and many of the letters never arrived, the same average six degrees already speculated were found.

The model of Watts-Strowgatz was designed to systematically study this phenomenon, by tuning some parameters. The exact model is the following: a set of N nodes is evenly ranged around a ring, and every node is connected to the closest $\frac{K}{2}$ nodes in each of the directions of the ring. N and K are such that $N \gg K \gg \ln N \gg 1$. We get then a regular lattice where every node has K neighbors. We take then every node, and we rewire each edge with probability $0 < \beta < 1$ to a different node selected randomly.

For a small β the model allows to have a very clusterized network -which is a typical characteristic of social networks-, but still be extremely well connected, since even a small increment in β dramatically reduces the average shortest paths of the network.

2.1.3.3 Barabasi-Albert

Although useful to understand some of the dynamics of real networks, a piece of empirical information was missing from the previous models: nor the Erdős-Rényi, nor the Watts-Strowgatz model generated graphs with a power law degree distribution, *i.e.* a degree distribution of the form $P(k) \sim k^{-\gamma}$. Now, although sometimes exaggerated [39], it is still widely agreed that this distribution is present in parts of many real networks. The presence of that distribution implies among other things that the number of nodes with high degree is quite large comparing with the Poisson distribution generated by the other models.

In 1999 Barabasi and Albert devised a mechanism that could at the same time have a natural interpretation, and generate a graph with a power law degree distribution [11]. For this, two new ingredients were added to the model: *growth* and *preferential attachment*.

For what concerns *growth*, the previous models were using a number N of nodes that didn't change over time. The Barabasi-Albert model proposed to start with a few nodes, and add more of them in discrete time steps.

Preferential attachment, on the other hand, formalizes the *"the rich gets richer"* notion, so nodes having a large degree are more likely to increase their degree even more.

The exact model is the following: we begin with a completely connected graph with m_0 nodes. We add new nodes one at a time, and we connect them with m of the existing nodes. Each new node is connected to a node i with a probability $p_i = \frac{k_i}{\sum_j k_j}$ -where k_i is the degree of the node-, so the larger the degree of a node, the bigger the probability for a new node of connecting to it.

The resulting graph has both a power law degree distribution, and a clustering degree (empirically calculated) much higher than a random graph.

These are just a few emblematic examples of dynamic random networks. All of them have been modified in many different ways to better account for real phenomena. Still they show very clearly the way of approaching the problem, where a building mechanism with a natural interpretation is proposed, some properties of the resulting graph are known, and the fact that the model generates these properties helps to confirm the intuitions behind the model proposed. This same kind of approach will be seen in the application part of the thesis, when we will talk about agent-based models.

2.1.4 Group Centrality

Although most network studies concentrate on properties of either some individual nodes, or the network as a whole, some work has been done to study groups of nodes. In particular Everett and Borgatti [57] generalized some centrality notions to a group of nodes. These group notions have not had the same impact in network studies than their individual counterparts, but we thought it important to have them in mind when will present pretopology, since that is also a framework for the study of group relations.

Everett and Borgatti decided to generalize only the canonical measures for the three metrics selected by Freeman: degree, closeness and betweenness. As a main guideline in their generalizations, they had in mind that it should keep the same metric when the group considered is a singleton.

2.1.4.1 Degree Centrality

The group degree centrality of a set C is defined as follows:

$$\begin{aligned} Group\ Degree\ Centrality &= |N(C)| \\ Normalized\ Group\ Degree\ Centrality &= \frac{|N(C)|}{|V| - |C|} \end{aligned}$$

Where $N(C)$ is the set of networks that have at least one neighbor in C.

The first thing that one should notice is that an external element connected to the group is only counted once. More than one connection to the same element is just redundant.

The idea of the normalization is to be able to compare among different groups. The authors justify the choice of denominator as that is the maximum size $|N(C)|$ can have. The normalization has as a consequence that among two sets with equal number of connexions, it is the bigger one that will be considered more central. This could make the choice of denominator look debatable, but it may be argued that the most central group will be the one that will cover more agents between the group and its connexions.

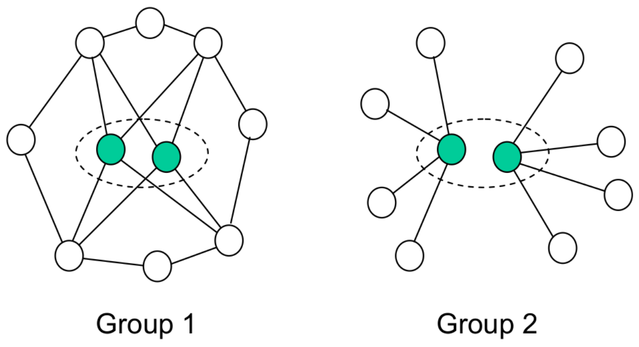


Figure 2.3: Group degree centrality
Two groups with the same sum of individual degrees, but different group degree.

2.1.4.2 Closeness Centrality

The group closeness centrality is defined as follows:

$$\begin{aligned} D_x(C) &= \{d(x,c),\ c \in C\},\ x \in V - C \\ D_f(x,C) &= f(D_x) \\ &Where\ f\ = min,\ max,\ mean,\ or\ meadian. \end{aligned}$$

$$Group\ Closeness = \sum_{x \in V-C} d_f(x,C)$$

$$Normalized\ Group\ Closeness = \frac{|V - C|}{\sum_{x \in V-C} d_f(x,C)}$$

In other words, D_x is the set of distances (*i.e.* shortest path lengths) from x to every member of C, and $D_f(x, C)$ is function that will take D_x and will transform it into a single number. This could be the minimum of the set, the maximum, the mean or the median. All of them will comply with the demand of being equal to the classical definition when singletons are considered.

So we can see that there is not a singular generalization of the concept, but a group of them. The choice of one particular distance is dependent on the problem we are trying to solve. If we are in a context where we model communication over edges, and we consider that the information arrives to the group as soon as one element of the group receives it, then the minimum distance is appropriate. If on the other hand, we need that all the nodes of the group receive the information before considering that the group is informed, then the maximum distance will be a better choice.

2.1.4.3 Betweenness Centrality

The extension of betweenness centrality is more straight forward than that of closeness. To calculate the centrality of the set C, we just need to calculate the number of shortest paths between pairs of nodes outside C and see how many of those pass through C. This extension, as that of the degree, is unique.

$$Group\ Betweenness\ Centrality = \sum_{s \neq t \mid \{s,t\} \cap C = \varnothing} \frac{\sigma_{st}(C)}{\sigma_{st}}$$

Where σ_{st} is the number of shortest paths that pass between s and t, and $\sigma_{st}(C)$, is the number of those paths that passes through C.

This metric can again be normalized by the maximum value the metric could take. This is $\frac{(|V-C|)(|V-C-1|)}{2}$. We get then:

$$Normalized\ Group\ Betweenness\ Centrality = \frac{2 \cdot \sum_{s \neq t \mid \{s,t\} \cap C = \varnothing} \frac{\sigma_{st}(C)}{\sigma_{st}}}{(|V-C|)(|V-C-1|)}$$

2.1.5 Applications

It is probably no longer necessary to try to convince about the great contributions that network science has done in almost every field of knowledge: physics, biology, linguistics, computer science, sociology. Networks are present at the heart of almost every system of transportation or communication used every day. As way of conclusion we will mention two examples among the myriad applications existent.

Besides the obvious connexions to the Internet and other local computer networks, where they play a key role in the design of the structure or the communication protocols, many other applications of networks can be found in the realm of computers. One relatively recent example is that of graph databases, where relational databases are now identified with graphs and queries are expressed using the semantics of the theory.

Networks have also played a key role in epidemic studies. An interesting example of this can be found in [42], where a network representing connections between susceptible people is presented and strategies of vaccination are tested. Turns out, the best random strategy consists in selecting random individuals, and then vaccinating a random neighbor of those individuals. The idea is that we are looking for the spreaders, those that have the largest number of connections, but it is not always evident who they are. By selecting someone randomly, and then selecting a neighbor, we are augmenting the probability of selecting a spreader, since they will be connected to most of the individuals that we may end up picking in the first place.

2.1.6 Multilayer Networks

A lot work has been done in the last decade on multilayer networks [49, 80, 5]. These are structures that consist of a set of regular networks, and potentially some connections between nodes on different networks. We can see a graphical representation in figure 2.4.

There have been three main uses for multilayer networks as a modeling tool:

- Each network is a snapshot in time of the relations in a same population.
- Each network represents different types of relations inside a unique population.
- Each network represents a different population.

Although many results have been found, they came from different fields, and there hasn't been a uniform way to deal with the subject. Even the name hasn't been unique: they have been referred as multiplex networks, interdependent networks, networks of networks and many others.

Some effort has been done recently [49] to generate a uniform framework to encompass all work done on multilayer networks. The authors propose to use tensor notation as a way to encode a multilayer network. They define the intra-layer adjacency tensor:

$$W^{\alpha}_{\beta}(\tilde{k}) = \sum_{i,j=1}^{N} w_{ij}(\tilde{k}) E^{\alpha}_{\beta}(ij)$$

Where N is the number of element in a singular network. This is the object describing the relations inside the layer $\tilde{k}$; the tilde being there to indicate that is the index of a layer. So for a multilayer network with a singular unweighted network, we would have $w_{ij}(\tilde{k}) = a_{ij}$, where a_{ij} is the i, j entry of the adjacency matrix.

To describe connections between layers the inter-layer adjacency tensor $C^{\alpha}_{\beta}(\tilde{h}\tilde{k})$ is introduced, which corresponds to $W^{\alpha}_{\beta}(\tilde{k})$ when $\tilde{h} = \tilde{k}$. So, if we denote the canonical basis $E^{\tilde{\gamma}}_{\tilde{\delta}}(\tilde{h}\tilde{k}) = e^{\tilde{\gamma}}(\tilde{h}) e_{\tilde{\delta}}(\tilde{k})$, we can write the multilayer adjacency tensor as:

$$M^{\alpha\tilde{\gamma}}_{\beta\tilde{\delta}} = \sum_{\tilde{h},\tilde{k}=1}^{L} C^{\alpha}_{\beta}(\tilde{h}\tilde{k}) E^{\tilde{\gamma}}_{\tilde{\delta}}(\tilde{h}\tilde{k})$$

Where L is the number of layers. As with the case of the adjacency matrix, the tensor representation is not only a formal way to encode a multilayer network, but also allows for some notions of spectral graph theory to be extended to its tensor counterparts [121]. In particular De Domino *et al.* [49] showed how to extend the notions of degree centrality, clustering coefficients, eigenvector centrality, modularity, Von Neumann entropy, and diffusion.

We finish this section mentioning two examples of application of multilayer networks:

- Buldyrev et al. [35] studied an interdependent network of power grids and computers, where the malfunctioning of nodes in one layer could affect the functioning of the nodes in the other layer: a power shutdown would alter the functioning of computers, and the computers played a role controlling the power grids. This model showed that this two layer version was less robust than a singular layer model, since fragile nodes in one layer could damage strong/important nodes in the other.

- Finn et al. [60] modeled a society of baboons using a two layer network, where each network encoded one type of relation: grooming and proximity. A comparison of the centrality of the baboons using this model with the centrality of the aggregated network, showed a difference in results. This is natural, since the aggregated version considers links in different networks to be redundant, when this is clearly not the case for that particular problem.

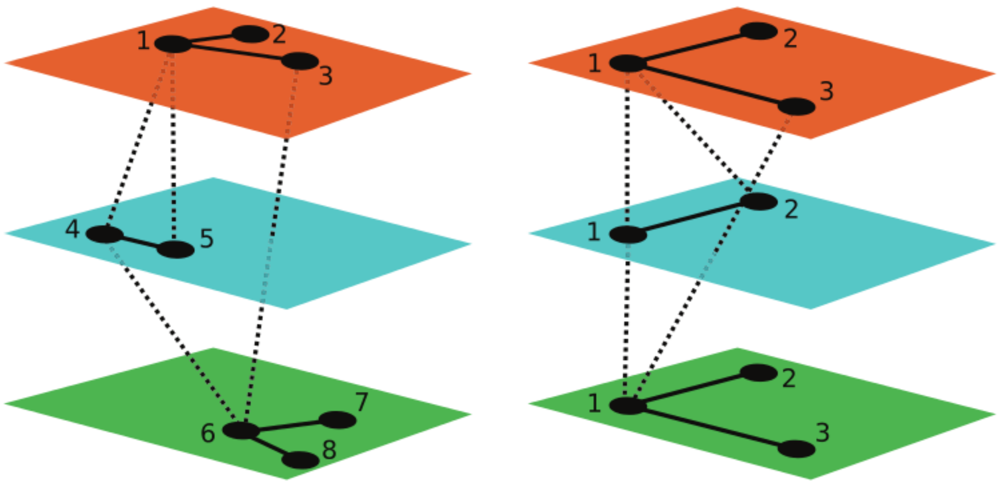


Figure 2.4: Multilayer network

2.1.7 Hypergraphs

For the sake of completeness we will mention the hypergraphs, although they have being significantly less used as a modeling tool. In a hypergraph edges are allowed to connect more than two nodes at the same time. So each edge is a subset of the elements of the system. Notions such as connectivity can also be extended, and the Laplacian hypergraph can also be used to extract information. It has been shown that any hypergraph can be identified with a particular pretopology [46].

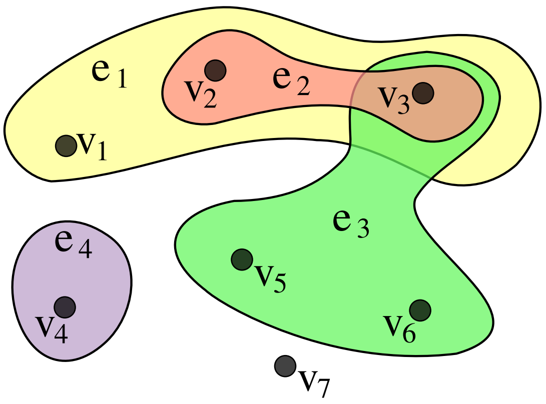


Figure 2.5: Example of an Hypergraph

2.2 Pretopology

In this section we give the most important definitions of the pretopology theory [?, ?, ?]. We explain what is a pretopological space, and we show two different ways of conceiving it. Throughout the whole section we mention how these different concepts could be applied to model problems from real life.

By the end of the section we should be able to understand what is it that pretopology can add as modeling tool, that was not already there in the previously mentioned theories.

2.2.1 Functional Definitions

Let's start by formally defining a pretopological space.

Definition 2.2. *A pseudoclosure function $a : \wp(U) \to \wp(U)$ on a set U, is a function such that:*

- $a(\emptyset) = \emptyset$
- $\forall A \mid A \subseteq U : A \subseteq a(A)$

Where $\wp(U)$ is the power set of U.

Definition 2.3. *A tuple $(U, a())$, where U is a set of elements and $a(.)$ is a pseudoclosure function on U, constitutes a pretopological space.*

We can see then that a pretopological space is defined by establishing a relation between any set of elements and a bigger set. We can immediately see how this could model situations where we are interested in studying a diffusion of information over a group of people; the application of the pseudoclosure to a set A would give us the set of people that was informed by A.

There is an alternative way of characterizing a pretopological space in terms of an interior function.

Definition 2.4. *An interior function $i : \wp(U) \to \wp(U)$ on a set U, is a function such that:*

- $i(U) = U$
- $\forall A \mid A \subseteq U : i(A) \subseteq A$

This second operator is related to the first one by a $c-duality$ relation, *i.e.* $\forall A \mid A \subseteq U : i(A) = a(A^c)^c$, where A^c is the complement of A. Although this second function can be used to model some phenomena of erosion or degradation, it is mostly the pseudoclosure function that has been used in applications. Since one can unambiguously define one in terms of the other, we will only concentrate our efforts on the description and discussion of the pseuclosure function.

The definition just given determines the most general pretopological space. By asking the function to fulfill some additional conditions we get more specific pretopological spaces:

Definition 2.5. *If $\forall A, B \mid A \subseteq U, B \subseteq U : A \subseteq B \Rightarrow a(A) \subseteq a(B)$, then we get a pretopological space of type V. This property is called isotony.*

Definition 2.6. *If $\forall A, B \mid A \subseteq U, B \subseteq U : a(A \cup B) = a(A) \cup a(B)$, then we get a pretopological space of type V_D.*

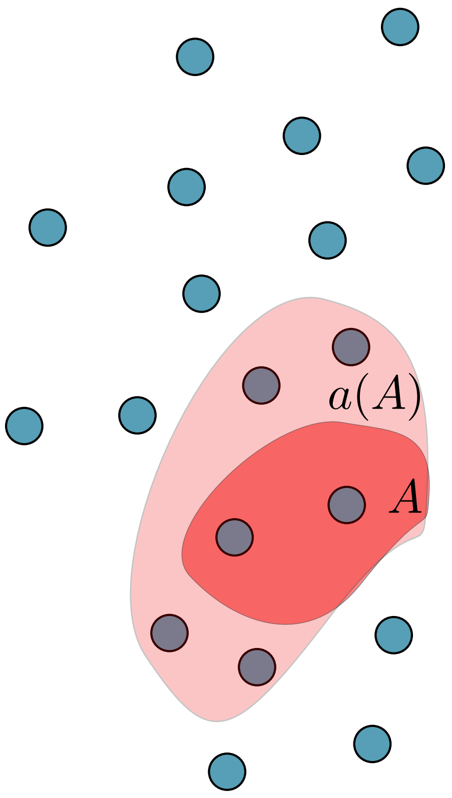


Figure 2.6: Example of a Pseudoclosure function

Definition 2.7. *If $\forall A \mid A \subseteq U : a(A) \subseteq a(A) = \bigcup_{x \in A} a(x)$, then we get a pretopological space of type V_S.*

Definition 2.8. *If we have a pretopological space of type V_D and $\forall A \mid A \subseteq U : a(A) = a(a(A))$, then we get a topology. The pseuclosure function here is said to be idempotent.*

It's interesting to realize that all these definitions are different degrees of relaxation of Kuratowski's axioms for a topology. This shows that pretopology is a generalization of the theory of Topology.

It's clear that in a finite space, $V_S = V_D$ [**?**]. Also, in pretopological spaces of type V_D the pseudoclosure of a set is completely defined by the pseudoclosures of its singletons. So if the space is also finite, we could draw an edge from an element to every element of its pseudoclosure, and the pseudoclosure would be equivalent to a particular graph. Figure 2.7 shows the relation between the two. This demonstrates that pretopology is also a generalization of graph theory.

Since the goal of the thesis is mostly to present the advantages of the theory of pretopology over other theories when modeling certain types of problems, we will not focus our attention into the particularities of V_D and V_S spaces.

2.2.1.1 Connectivity

Once we have a pretopological space defined, we can study which sets can be reached by which others by iteratively applying the pseudoclosure function. The closest thing to this in a graph is the study of connected components, but the pretopological operators make possible to define a larger set of concepts. We will present some of them here, while the exact equivalent of the connected components will be presented in section Structural Equivalence 5.2.

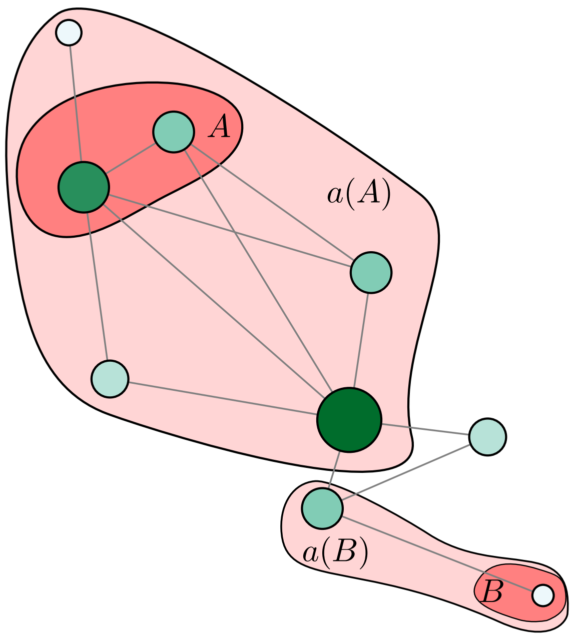


Figure 2.7: Pseudoclosure function on a graph.

There is a natural way of passing from a graph to a pretopological space of type V_D on a finite set, and vice versa.

Definition 2.9. *Given a pretopological space $(U, a(.))$, any subset A of U is said to be a closed subset of U if and only if $A = a(A)$*

Definition 2.10. *Given a pretopological space $(U, i(.))$, any subset A of U is said to be an open subset of U if and only if $A = i(A)$.*

Without going into much detail, let us give a simple example of how pretopology could be used as a modeling framework, and what would these definitions teach us. Let us suppose again that we are using pretopology to model information diffusion over a population of individuals, so the pseudoclosure $a(A)$ of a set A determines how the information spreads from A at time t, to $a(A)$ at time $t + 1$. A closed set B would be a set that doesn't spread information anymore. An open set O, on the other hand, is a set that will not receive the information from the exterior.

A concept related to that of *closed set*, that we use many times over the thesis, is that of a *closure*:

Definition 2.11. *Given a pretopological space $(U, a(.))$, we call closure of any subset A of U, when it exists, the smallest closed subset of $(U, a(.))$ which contains A. The closure of A is denoted by $F(A)$.*

The closure can also be defined as the intersection of all closed sets that contain A.

It is important to realize that the closure does not necessarily exist. When thinking of the closure as the smallest closed set, this would mean there are two or more closed sets containing A, without any containing the other, and without containing a smaller closed set bigger than A. When thinking of the closure as the intersection of all closed sets, this would mean the intersection is not closed. Table 2.1 shows an example of a pretopological space where not every set has a closure. Indeed, the singleton $\{x\}$ is contained by the closed sets $\{x, y\}$ and $\{x, z\}$, the intersection of which is again the singleton $\{x\}$. But we know $\{x\}$ is not closed, so the set does not have a closure.

The following property about closures is very important:

Proposition 2.1. *In a pretopological space of type V, every set has a closure. The proof can be found in [?]*

In a pretopological space of type V we can find the closure by repeatedly applying the pseudoclosure operator to the set and its subsequent images until it stops expanding. We can see an example of this in figure 2.8.

Pseudoclosures and Closures								
$\wp(U)$	$\varnothing$	$\{x\}$	$\{y\}$	$\{z\}$	$\{x,y\}$	$\{x,z\}$	$\{y,z\}$	U
$a(.)$	$\varnothing$	$\{x,z\}$	$\{y\}$	$\{z\}$	$\{x,y\}$	$\{x,z\}$	U	U
$F(.)$	$\varnothing$	___	$\{y\}$	$\{z\}$	$\{x,y\}$	$\{x,z\}$	U	U

Table 2.1: Example of pseudoclosures and closures on a space U

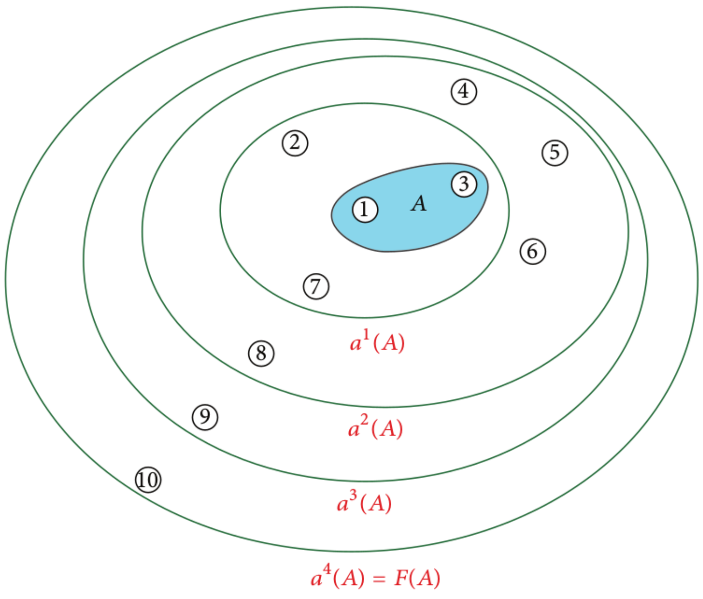


Figure 2.8: Closure of set A

Closure of a set found by iteratively applying the pseudoclosure operator.

2.2.2 Neighborhood Duality

There is a second way of characterizing pretopologies of type V and V_D. To understand it we need to give a few more definitions first:

Definition 2.12. *We say that a set $\mathcal{F}$ of $\wp(\wp(U))$ is a **prefilter** over U, if:*

$$\forall F \in \mathcal{F}, \forall H \in \wp(U), F \subset H \implies H \in \mathcal{F}$$

Definition 2.13. *We say that a set $\mathcal{F}$ of $\wp(\wp(U))$ is a **filter** over U, if it's a prefilter stable under finite intersection, i.e.:*

$$\forall F \in \mathcal{F}, \forall G \in \mathcal{F}, F \cap G \in \mathcal{F}$$

In other words, and restricting ourselves to a finite space, a **filter** is the family of all supersets of a set B, while a **prefilter** is the family of supersets of every member B_i of

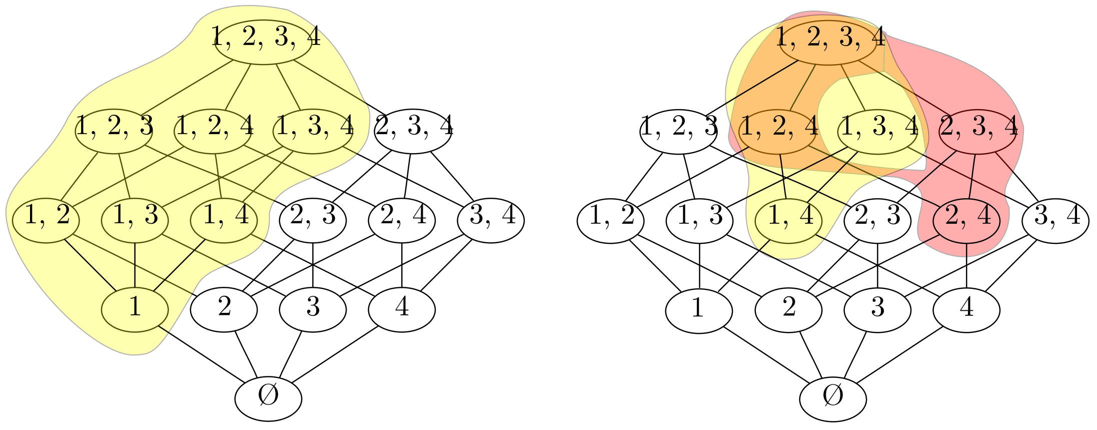


Figure 2.9: Filters versus Prefilters

On the left we see a filter, on the right a prefilter with basis $\mathcal{B} = \{\{1,4\}, \{2,4\}$.

a family of sets $\mathcal{B}$. The family of sets $\mathcal{B}$ is called the **basis of the prefilter**. We can see in figure 2.9 an example of a filter and a prefilter with basis $\mathcal{B} = \{\{1,4\}, \{2,4\}$.

Now, if we have a set U, and for ever $x \in U$ we have a prefilter $\mathcal{V}(x)$ such that every member of $\mathcal{V}(x)$ contains the element x, we can define a pseudoclosure function in the following way:

$$\forall A \subseteq U, \quad a(A) = \{x \in U \mid \forall V \in \mathcal{V}(x), V \cap A \neq \emptyset\}$$

We call the prefilter $\mathcal{V}(x)$ the **family of neighborhoods** of x, and each set in the family is called a **neighborhood** of x. Figure 2.10 shows a graphical representation of this definition of the pseudoclosure.

On the other hand, if we have a pseudoclosure function $a(.)$ in a pretopological space of type V, the family of sets given by:

$$\mathcal{V}(x) = \{V \subset U \mid x \in i(V)\}$$

where $i(A) = a(A^c)^c$, is a prefilter.

The following proposition shows that we can go from one definition to the other interchangeably:

Proposition 2.2. *No two families of prefilters $\{\mathcal{V}(x_i) \mid x_i \in U\}$ define the same pseudoclosure function $a(.)$, and no two pseudoclosure functions define the same family of prefilters $\{\mathcal{V}(x_i) \mid x_i \in U\}$.*
The proof can be found in [?]

This duality is the single most important property of a pretopology when it comes to use it as a modeling tool. Although not a clear cut division, the two ways of conceiving a pretopological space invite to model different kinds of phenomena:

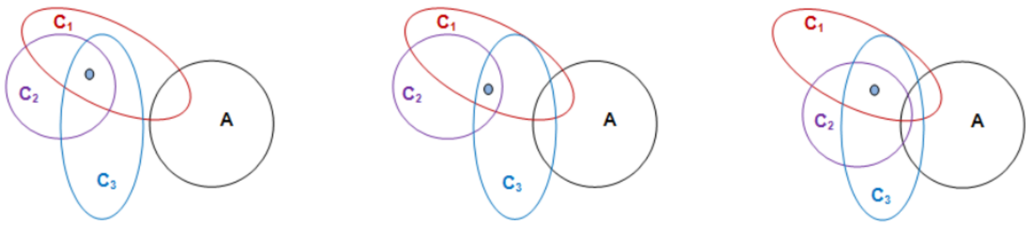
- By looking at a set, and asking what is its pseudoclosure, we tend to think about the way a group expands, and we usually think of dynamic phenomena of propagation or expansion.
- By asking to each external element if its neighborhoods intersect a set A, we are positioning ourselves from the perspective of the external element. We are asking every element if it belongs or not to the pseudoclosure. This process invites us to model relations existent between an element and a group, and to conceive the whole space as a static structuring framework.

So in the same way that a single directed weighted network could be used to study a dynamical problem like the flow of a liquid in a structure, and also a static problem, like the influence of different individuals in a population; also a single pretopological space could be used to model the dynamics of a propagation or the static structure of a group, depending on the interpretation we give to the elements of the space.

Let us look at two simple examples of relations between an element and a group to better grasp the utility of this second interpretation:

- Let us say we are interested in modeling the topical relation of a word to a group, *i.e.* a word will be related to a group of words if they all belong to the same topic. For example, *racket* could be related to the set {*ball*, *sport*}, but we may not want it to be related to {*ball*, *sport*, *basket*}.
- If we are modeling the votes that a presidential ticket would get, asking to each element if it belongs to the pseudoclosure of a pair {president, vice-president} to model a vote for the ticket, is also a fundamental element-group relation. Indeed, those voting for a ticket are not necessarily those voting for one or the other candidate, since the candidates can either strengthen each other or generate rejection.

This two examples show relations from an element to a group that are different from the sum of the individual relations. This is what makes them so difficult to be modeled by a graph. On the other hand, each element entertains a single relation with the group, so multilayer networks do not seem to be apt for the task either.



(a) $x \notin a(A)$ (b) $x \notin a(A)$ (c) $x \in a(A)$

Figure 2.10: Neighborhood definition of a pretopology

2.2.3 Generalizing the Neighborhoods

The way of defining the image $a(A)$ of a pseudoclosure function $a(.)$ from a set of neighborhood families, was to ask every single neighborhood of an element x to intersect the set A.

Now, that definition might be too restrictive when one is trying to use neighborhoods to define a pseudoclosure, so some generalizations have emerged. The first of this generalizations was to define the concept of a *weak pretopology*, as opposed to a *strong pretopology*, which is the name now given to the original definition.

On a set U where every $x \in U$ has a prefilter of neighborhoods $\mathcal{V}(x)$, the pseudoclosure function of the *weak pretopology* is given by:

$$\forall A \subseteq U, \quad a(A) = \{x \in U \mid \exists V \in \mathcal{V}(x), V \cap A \neq \varnothing\}$$

So instead of asking to all neighborhoods to intersect the set, we only ask for one neighborhood to do it.

It is important to realize that under the *weak pretopology* we no longer have a unique way of generating a family of prefilters. For example, let us consider these two families of prefilters $\mathcal{V}_1$ and $\mathcal{V}_2$:

$$\begin{aligned} V_1(a) &= \{\{a,b\},\{a,c\},\{a,b,c\}\} \\ V_1(b) &= \{\{a,b\},\{b,c\},\{a,b,c\}\} \\ V_1(c) &= \{\{a,c\},\{b,c\},\{a,b,c\}\} \end{aligned}$$

$$\begin{aligned} V_2(a) &= \{\{a,b,c\}\} \\ V_2(b) &= \{\{a,b,c\}\} \\ V_2(c) &= \{\{a,b,c\}\} \end{aligned}$$

For both these families the *weak pretopology* generated is the one that has the whole set as the pseudoclosure of every subset of the space.

This proves that we no longer can go back from the pseudoclosure to the prefilter family. In practice, this has not really been a problem, since most of the time we are trying to model a system, we are interested in going in the other direction.

Through the thesis we will see many more extensions to the way of building a pseudoclosure from the neighborhoods of a set.

2.2.4 Pretopological Operators

Now that we have seen what a pretopological space is, and how can it be used to represent a system, let us see some additional concepts and metrics that can be defined in it.

We will begin by studying some pretopological operators, *i.e.* some functions we can apply to the subsets of the space. We have already seen two of them, the pseudoclosure $a(.)$ and the interior $i(.)$; six other operators can be defined:

Definition 2.14. *$\forall A \in \wp(U)$, we call the rim of A, denoted $b(A)$, the function:*

$$b(A) = \{x \in A \mid \forall B \in \mathcal{B}(x), B \cap A^C \neq \varnothing\}$$

Definition 2.15. *$\forall A \in \wp(U)$, we call the girdle of A, denoted $o(A)$, the function:*

$$o(A) = \{x \in A^C \mid \forall B \in \mathcal{B}(x), B \cap A \neq \varnothing\}$$

Definition 2.16. *$\forall A \in \wp(U)$, we call the frontier of A, denoted $f(A)$, the function:*

$$f(A) = \{x \in U \mid \forall B \in \mathcal{B}(x), B \cap A \neq \varnothing \wedge B \cap A^C \neq \varnothing\}$$

Definition 2.17. *$\forall A \in \wp(U)$, we call the derivative of A, denoted $d(A)$, the function:*

$$d(A) = \{x \in U \mid \forall B \in \mathcal{B}(x), (B - \{x\}) \cap A \neq \varnothing\}$$

Definition 2.18. *$\forall A \in \wp(U)$, we call the coherence of A, denoted $c(A)$, the function:*

$$c(A) = \{x \in A \mid \forall B \in \mathcal{B}(x), (B - \{x\}) \cap A \neq \varnothing\}$$

Definition 2.19. *$\forall A \in \wp(U)$, we call the exterior of A, denoted $e(A)$, the function:*

$$e(A) = \{x \in U \mid \exists B \in \mathcal{B}(x), B \subset A^C\}$$

The utilization of these operators has mainly concerned the field of imagery [82, 22], where the sets resulting from the different applications of the functions have served to detect the outline or silhouette of a figure, or even to improve the resolution of an image.

The following proposition will be useful for us:

Proposition 2.3. *$\forall A \in \wp(U)$, we will call the exterior of A, denoted $e(A)$, the function:*

$$\begin{aligned}
a(A) &= A \cup d(A)\\
i(A) &= (a(A^C))^C\\
b(A) &= A - i(A) = A - (a(A^C))^C\\
o(A) &= a(A) - A\\
f(A) &= b(A) \cup o(A) = (A - (a(A^C))^C) \cup (a(A) - A)\\
c(A) &= A \cap d(A)\\
e(A) &= a(A)^C
\end{aligned}$$

We can see then that knowing the pseudoclosure gives us access to all other operators. Only the derivative cannot be immediately formulated as a function of the pseudoclosure and the set, but we can still express it as the following relation: $d(A) = a(A) - \{x \in A \mid \exists B \in \mathcal{B}(x) B \cap A = x\}$. In other words, we need to take from the pseudoclosure A, those elements inside A that have the rest of a neighborhood completely outside of A.

During the rest of the thesis we will concentrate only in the pseudoclosure function. First, because of these direct relations to the other operators, but more importantly, because it is by far the most employed operator when using pretopology as a modeling tool.

2.2.5 Metrics

Just as graph theory proposes some metrics of centrality that allow us to extract information from the graph representation of a system, in [91] Levorato introduced a group generalization of degree, betweenness and closed centrality, using the concepts of pretopology.

The first definition we will see, is that of group degree centrality:

Definition 2.20. *We define the **group degree centrality** of a set $A \in \wp(U)$ in a pretopological space $(U, a(.))$, denoted $C_d^G(A)$ as:*

$$C_d^G(A) = \left| a(A) - A \right|$$

This notion is similar but not equal to the one proposed by Everett and Borgatti [57]for the case of a pretopology in a graph. Indeed, although both count only once each node connected to the elements of the group, Levorato prefers not to normalize by the maximum possible value. He estimates that the bias introduced towards bigger sets is not justified from a modeling point of view.

Definition 2.21. *We define the **group betweenness centrality** of a set $A \in \wp(U)$ in a pretopological space $(U, a(.))$, denoted $C_b^G(A)$ as:*

$$C_b^G(A) = \sum_{x}^{|U|} \sum_{y \neq x}^{|U|} \phi(A)$$

$$\phi(A) = \begin{cases} 1, & \text{if } \exists p_{x,y}(A) \\ 0, & \text{otherwise} \end{cases}$$

and,

$$\exists p_{x,y}(A) = \textbf{True} \iff \exists A \in \wp(U) \mid A \subseteq a^k(\{x\}) \wedge \{y\} \subseteq a^r(\{x\}) \wedge k \leq r.$$

In other words, the betweenness centrality of a group A will be the count of the number of paths between pairs of singletons that pass through A. And we will say that a path from $\{x\}$ to $\{y\}$ will pass by A if the continuous applications of the pseudoclosure operator cover the whole set A before reaching the singleton $\{y\}$. An example of this can be seen in Figure 2.11.

One difference with the definition of Group Betweenness Centrality given in the graph section, is that we do not need to calculate the fraction of the shortest paths that pass through the set, since only one path is considered between the singletons. This makes that the definition will not have the same result as the traditional betweenness centrality, if we apply it to a singleton on a pretopological space built over a singular graph. This metric is not a generalization then, but a fundamentally different way of conceiving the concept.

Definition 2.22. *We define the* ***group closeness centrality*** *of a set $A \in \wp(U)$ in a pretopological space $(U, a(.))$, denoted $C_c^G(A)$ as:*

$$C_c^G(A) = \sum_k \frac{|a^k(A) - a^{k-1}(A)|}{k}$$

The sum is carried out until the closure is reached.

This metric will not be a generalization either.

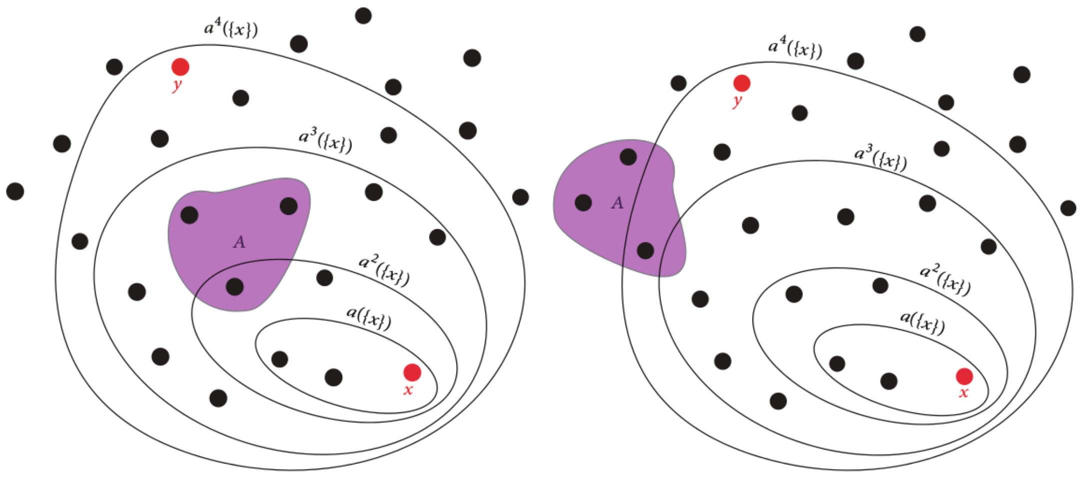


Figure 2.11: Pretopological path

It is important to realize that these metrics are fundamentally pretopological, so although we have been comparing the difference with previous measures of group centrality on a graph, they can be applied to a much wider range of spaces.

2.2.6 Applications

The examples we have given so far have tried to show how pretopology allows to model relations between an element and a group. Our main interest was to show its applicability when the relations existing among groups of agents are not just the sum of the individual relations. These have been examples where the relation between an element an the group has been imposed by the model maker, because they have a natural interpretation.

Sometimes the relation will be described as a condition the element needs to have in relation to the group. For example, Dalud-Vincent defines in [44] a pretopological space over a set of authors of scientific papers, where an author belongs to the pseudoclosure of a group if he/she was the first author of a paper and all of his/her co-authors are inside the group.

Pretopology will be useful in other cases too, when the relation between the element and the group emerges from a combination of multiple types of relations. For example, in [91] Levorato studies the possible coalitions among the monks of a convent, where different types of relations between the monks have been quantified. This way, he can define that a monk joins a coalition if the sum of the esteem he has for the members of the coalition is bigger than a threshold, and if the sum of the disesteem the members feel for him is less than another threshold. This is just one example of the many possible ways to mix binary links to get a new relation between an element and a group.

Sometimes we do not know exactly what relation we should define, but we know some other properties of the pretopological space that we want to specify. This is the idea behind the concept of Learning a Pretopologcal Space, introduced by [41].

Some other applications of pretopology have been in the areas of classification [7, 63, 64, 2], clustering [87, 32, 29], complex network analysis [30, 91, 111, 81, 89, 29, 31], economic analysis [8] and image analysis [82, 22, 99].

Many of these examples are treated in detail during the thesis.

2.2.7 Conclusion

In conclusion we could say:

- Pretopology can be used to represent a system where the relation between an element and a set is not a simple aggregation of the individual relations to the members of the set. In this it is fundamentally different from a graph.
- Pretopology establishes one single relation between a particular element and a particular group. In this it is different from a multilayer network.

Hopefully we will be convinced about the utility of theory as a modeling framework. On the other hand, nothing has been said so far about its practical implementation. Now, pretopology was born as a practical theory [**?**], but its functional definition is prohibitively expensive even for small sets.

This has caused the neighborhood approach to become more popular, but the generalizations of this notion have been done in an ad-hoc manner. Each work defining its own rules and beginning from scratch.

In the next chapter we present a unified framework that tries to change that.

Part II

Algorithmic Contributions to the Theory of Pretopology

Chapter 3

Pretopology Formalization

This is the central chapter of the thesis. We present here a framework to formalize a pretopological space other than declaring the image of the pseudoclosure function for every subset of the space. When building the framework we had the following goals in mind:

- Homogenize the different practical ways of building a pretopology into a single framework that will cover them all.
- The framework should be operational. In particular, a pretopoly described using our framework should be economically stored in a computer.
- The different algorithms should be easily translated into our framework in order to be implemented in a computer.

The following chapters use this framework when talking about a pretopological space: first to study the complexity of some algorithms; then to implement and apply the pretopological concepts to some real problems.

3.1 Framework Description

In this section we present the framework that we use to describe a pretopological space, and we justify its choice.

The need for a framework comes from the exponential growth of the subsets of a set. If we were to use a pretopology over a set U in the way that is normally defined, the description of the space would demand to store every single image under the pseudo-closure function, for every possible subset of U. Now, it is well known that the number of subsets of a set with n elements is 2^n. By way of example, even for a set U with as few as 30 elements, the number of different images to store would be bigger than the number of bytes in a terabyte.

It is obvious then that any hope to use the concepts of pretopology out of the realm of mathematics would need to treat the subject differently. As it was briefly mentioned in the introduction, most of the works that have used pretopology as a conceptual framework have chosen to work with some generalization of the neighborhood concept. Instead of defining the image of the pseudoclosure function, we define rules to decide if an element belongs or not to the pseudoclosure of a particular set, and we calculate the image when it is necessary. We will see in the next section how diverse has been the landscape of practical ways to build a pretopology.

This lack of uniformity in the kind of rules that have been used has hindered the possibility of building in the works of others. Although concepts are shared and inherited from one work to the other, implementations remain in charge of the researcher and must be done from scratch every time: this prevents the subsequent improvements of the code and keeps researchers that are less comfortable with programming from using the concepts of the theory.

The problem may also be theoretical; a researcher needs to demonstrate every time from the beginning, if the rules he or she has defined give origin to a pretopolgy of type V or V_D.

A final difficulty steaming from this absence of framework is the impossibility to deal with complexity issues. Indeed, many pretopological algorithms [44, 23, 83] take as input a pretopological space, assuming a ready knowledge of the pseudoclosures of all elements. A study of the complexity of the algorithm in terms of the size of the input would not be very useful in that situation, since the input itself would become quickly inaccessible.

The goal is then to devise a more economical way to store and treat pretopologies, without having to get rid of its concepts.

Before presenting the framework, we remind that we are only interested in pretopologies with a finite number of elements. This should not be a problem, since our main goal is to implement the different pretopological metrics and algorithms and use them as a tool for applications where the elements will be discrete and finite.

Each pretopological space will be characterized by a tuple $(\mathcal{N}, \Theta, DNF(.))$, where:

- $\mathcal{G} = \{G_1(V_1, E_1), G_2(V_1, E_1), ..., G_n(V_n, E_n)\}$ is a set of n weighted directed graphs.
- $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$ is a set of n thresholds, each associated to one graph.
- $DNF(.) : (\wp(U), U) \to \{\textbf{True}, \textbf{False}\}$, where $\wp(U)$ is the power set of U, is a boolean function expressed as a positive disjunctive normal form in terms of the n boolean functions $V_1(A, x), ..., V_n(A, x)$, each associated to a graph, and whose truth value depends on the set A and the element x.

And we will determine if an element $x \in U$ will belong to the pseudoclosure of a set A in the following way:

- $\forall V_i(A, x),\ V_i(A, x) = True \iff \sum_{e_{xy} \in G_i,\ y \in A} w(e_{xy}) \geq \theta_i$, where e_{xy} is the edge going from x to y, and $w(e)$ is the weight of the edge e.
- The element $x \in U$ will belong to the pseudoclosure of $A \iff$ the $DNF(.)$ evaluates to **True**.

Simply put, we check for every graph if the sum of the weights of the edges going from the element x to the elements inside A is bigger than the threshold associated to the graph. When that happens, the boolean variable associated to that graph gets a value of **True**, otherwise it gets a value of **False**. If $DNF(.)$ evaluates to **True** with those values for the boolean functions $V_i(A, x)$, then the element belongs to the pseudoclosure. We can see an example of this in figure 3.1

Sometimes we will use $x \in a_i(A)$ as another way of saying that $V_i(A, x) = \textbf{True}$.

We should be careful with the direction of the edges of the graph. We are summing the edges that go from an element x to a set A. We have done this guided by the notion of asking to an element how strong is its connection to a group before deciding to add it. On the other hand, if we wanted to study a directed graph using a pretopology, the direction of the arrows should be changed in order to preserve the same connectivity notions as graph theory. This is achieved simply by transposing the adjacency matrix.

Although the framework is quite simple, we try to prove through the next sections that it is actually very powerful.

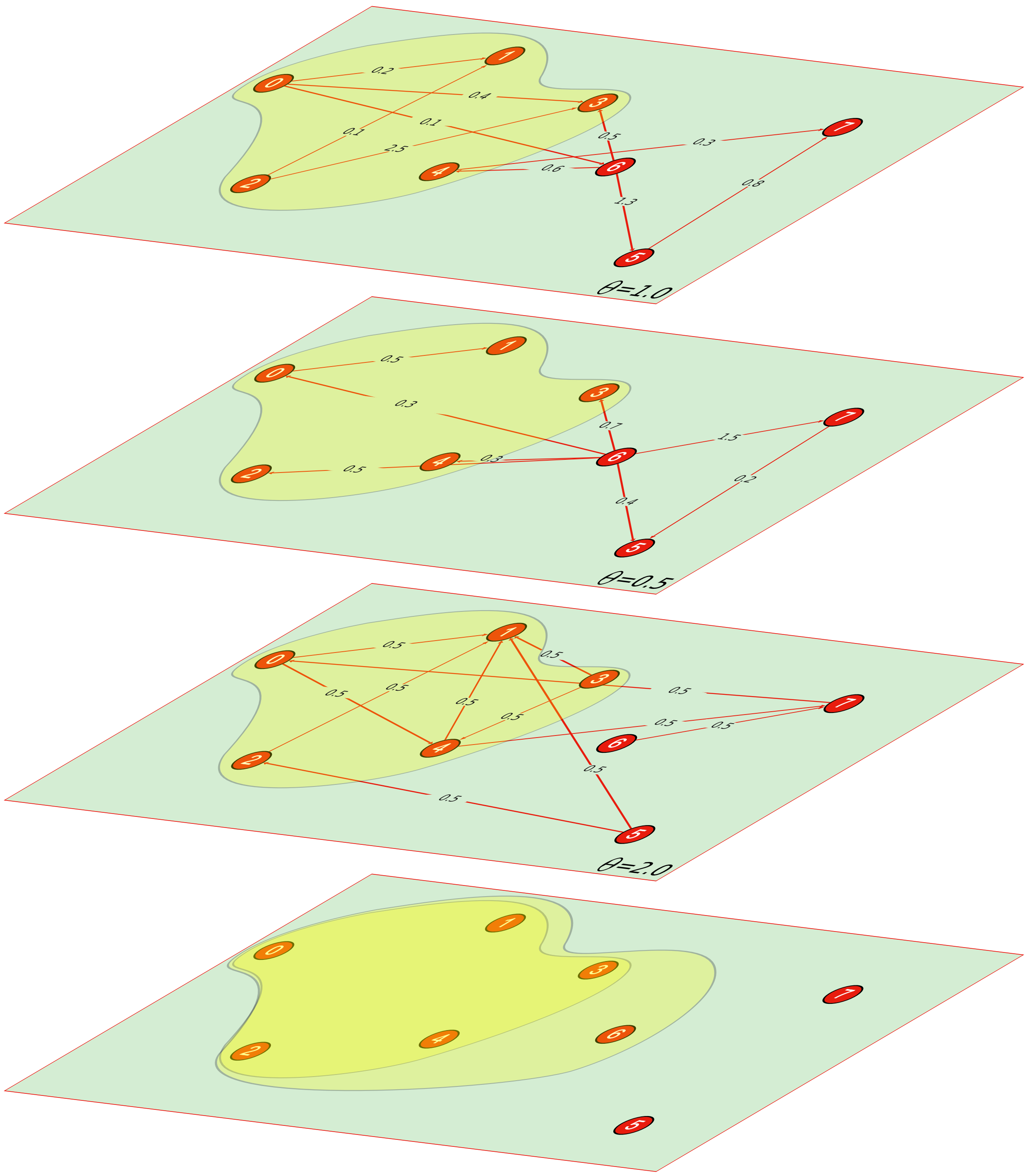


Figure 3.1: Example of pseudoclosure under the framework

Example of the pseudoclosure of a set A, on a pretopology defined by three graphs $G_1, G_2 and G_3$, thresholds $\theta1 = 1.0, \theta2 = 0.5$ and $\theta3 = 2.0$, and $DNF(A, x) = V_1(A, x) \vee (V_2(A, x) \wedge V_3(A, x))$.

We first prove its generality by showing how a great number of previous uses of pretopology as a modeling tool fit into this framework.

Secondly, we profit from the fact that the framework is based on networks, in order to transform some pretopological notions into graph or integer linear programming questions. This has multiple advantages: from a theoretical point of view, we are connecting the theory to two very active research programs, which can make the evolution of the field a lot more dynamic; from a more practical point of view, not only we can use the techniques from those fields to answer pretopological questions without the need to reinvent the wheel, but we can also use the multiple softwares and libraries that have been developed in those areas.

Finally, by assigning specific data structures to a pretopology, we are able to study the complexity of the algorithms that have been proposed so far, and to better analyze some ways of improving them.

This whole approach could be compared to the difference between networks and graphs. It is a pragmatic way to deal with the subject, where we will be able to systematically characterize families of pretopologies for which some algorithms can be used with great chances of success. Indeed, some of the algorithms commonly used in network theory may not have a good theoretical answer for every possible graph, but have been proven to perform very well for certain families of graphs commonly found in practice [48].

3.2 Equivalences

In a recent state of the art Bui [29] identified six practical ways of defining a pretopology. We show here how each of these procedures can be easily transformed to fit into our framework. Then, we take some other pretopologies defined in the literature which are not included in the previous cases, and we show that although very specific at first sight, they can still can be accommodated into our scheme. Finally we present a generalization of this last idea that allow us to express any possible V-space by means of our framework.

3.2.1 Practical ways to build a pretopological space

3.2.1.1 Pretopological space built from a basis of neighborhoods

This is the original formulation, the one that was presented in the previous chapter. We remind that each element has a prefilter basis associated to it, and it will belong to the pseudoclosure of a set iff all the elements of the basis intersect the set.

Since we are working in a finite space, the number of elements belonging to a particular basis set is also finite. Let us say each element $x_i \in U$ has n_i different sets in its prefilter basis: $B_{i\_1}, B_{i\_2}, ..., B_{i\_n_i}$.

Now let us define $n_{max} = \max_{x_i \in U} n_i$, the maximum number of different sets in the prefilter basis of an element. We can then create n_{max} different graphs, where an edge exists from x_i to y in the jth graph G_j, iff $y \in B_{x_i\_j\%n_i}$. We should notice that the way in in which we enumerate the elements of the basis is not important.

In other words, an element is connected to another in the jth network, if the other element belongs to the jth set of the basis of its prefilter. If j is bigger than the number of sets in the prefilter basis of the element, then we start over. By doing this a number n_{max} of times, we are sure all sets of all basis have been covered. That last step may seem strage; after all, one may be tempted to stop adding links if we already covered all the elements of the basis; but this would create an isolated node on the pretopology, since we ask the neighbors of every network to intersect the set.

If we consider all edges of all networks to have a weight equal to 1, and we associate the threshold $\theta = 1$ to every network, then the $DNF(.) = V_1(.) \wedge V_2(.) \wedge ... \wedge V_{n_{max}}(.)$ will get us an equivalent pretopology to the one defined by the prefilters.

Indeed, an element x will belong to the pseudoclosure of a set A iff it is connected with at least one edge on every network, and this will happen iff there is an element of every set of the basis inside A, which means that every neighborhood intersects the set.

It is obvious that to get the weak pretopology, where only one element of the basis had to intersect, we only need to switch the $DNF(.)$ from $DNF(.) = V_1(.) \wedge \ldots \wedge V_{n_{max}}(.)$ to $DNF(.) = V_1(.) \vee V_2(.) \vee \ldots \vee V_{n_{max}}(.)$.

3.2.1.2 $\mathbb{Z}^2$

This is just a particular case of the previous one. Here the elements of the set U are the points $(x, y) \in \mathbb{Z}^2$, and the basis are usually sets of immediate neighbors of the point. The pseudoclosure of A, as usual, asks for every member of the basis of an element to intersect A, in order for the element to be part of it.

Two of the most classical examples of pretopological spaces are those defined by the Von Neumann neighborhood (B_4), and the Moore neighborhood (B_8):

$B_4((x, y)) = \{(x + 1, y), (x - 1, y), (x, y), (x, y + 1), (x, y - 1)\}$

$B_8((x, y)) = \{(x + 1, y), (x + 1, y - 1), (x, y - 1), (x - 1, y - 1), (x - 1, y), (x - 1, y + 1), (x, y + 1), (x + 1, y + 1), (x, y)\}$

It is important not the confuse neighborhoods and basis. B_4 and B_8 are neighborhoods, *i.e.* each of them is the single set that belongs to the basis of their space.

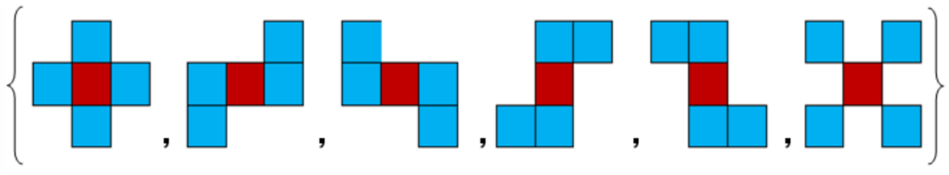
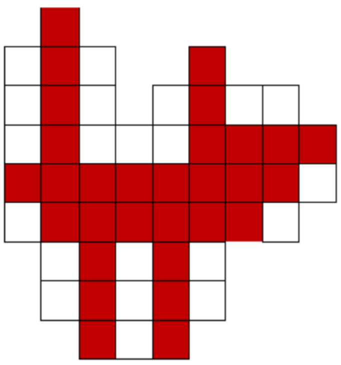
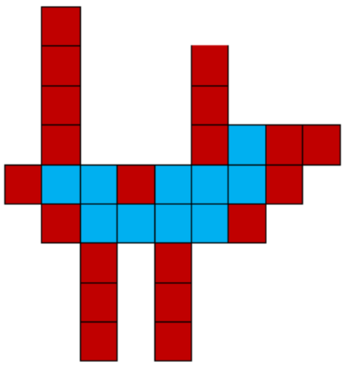


Figure 3.2: Example of a basis in $\mathbb{Z}^2$



(a) A in red and a(A) box



(b) A in red and i(A) in blue

Figure 3.3: Pseudoclosure of a set with the previous basis

There are two particularities that make this kind of pretopology deserve some attention:

- The first one is that it has many applications in imagery, by identifying the points with the pixels of an image [82, 22]. The different pretopological notions can then be applied to parts of the image in order to get a better resolution or to segment it.

- A second one, related to our framework, is that every set has the same type of neighborhoods, so the resulting networks are very symmetrical. We can see the networks generated by the basis containing the Von Neumann and the rotated Von Neumann neighborhoods in figure 3.5.

In order to completely define the pretopological space the weights of the networks' edges, and the thresholds should all be set to 1. The $DNF(.)$ is defined as usual with the prefilter case, $DNF(.) = V_1(.) \wedge V_2(.) \wedge ... \wedge V_{n_{max}}(.)$.

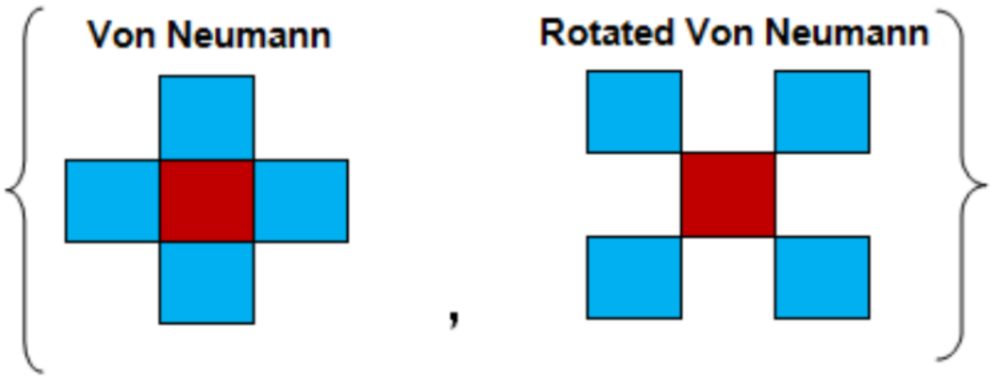


Figure 3.4: Von Neumann and rotated Von Neumann neighborhoods

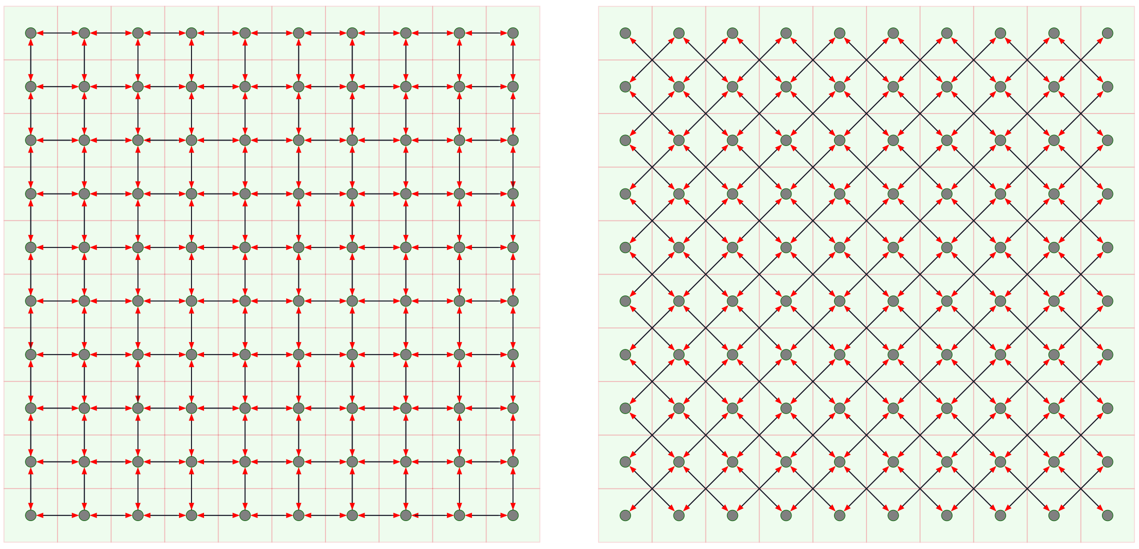


Figure 3.5: Graphs obtained from the Von Neumann and rotated Von Neumann neighborhoods

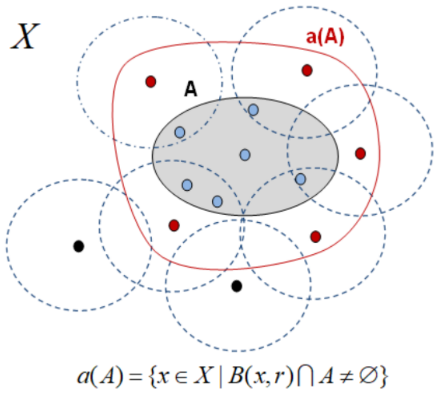
3.2.1.3 Pretopology in a metric space

Let us consider a set U endowed with a metric $d(.)$, and a positive real r. By considering the balls with center $x \in U$ and radius r:

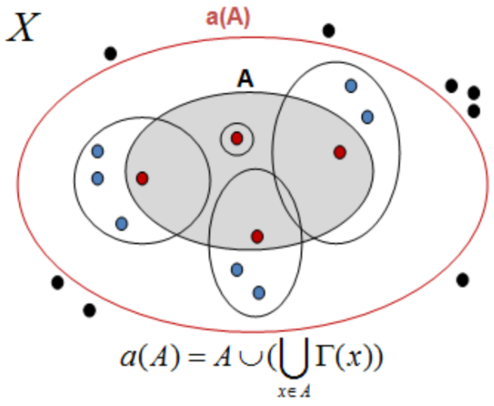
$$B(x,r) = \{y \in U| \, d(x,y) \leq r\}$$

we can define a pretopological space by setting $\{B(x,r)\}$ as the prefilter basis of the element x.

We can see then that this is again a particular case of the prefilter basis definition of a pretopology. Here we actually have a filter, since the basis have always one element. Just as in the $\mathbb{Z}^2$ case, the basis are similar (not the same, of course) for every element, so if we had our elements arranged in the form of a grid we could recover the previous case with the Von Neumann basis for a particular value of r. We recall from the context section that this is a pretopological space of type V_D, and since we are working with a finite number of elements it is also a type V_S. V_D spaces on a finite space, as we already said, are equivalent to a network, so it is obvious that this fits into our framework. Indeed, $\forall x$ the number of elements that will be closer than r will be finite, and we can always create an edge from x to them.



(a) a(A) in a metric space



(b) a(A) in a space equipped with a neighbor function

Figure 3.6: Examples of pseudoclosure

3.2.1.4 Pretopology in a space equipped with a neighbor function

A neighbor function is a multivalued function $\Gamma : U \rightarrow \wp(U)$, where $\Gamma(x)$ is the set of neighbors of the element x. We define the pseudo-closure $a(.)$ as follows:

$$\forall A \in \wp(U), a(A) = A \cup (\bigcup_{x\in A} \Gamma(x))$$

Now, since we are working in a finite space, the multivalued function can be uniquely identified with a directed graph, where an edge exists from x to y if y belongs to $\Gamma(x)$. This is actually how Claude Berge defined a graph.

To get the equivalent pseudoclosure inside our framework we will take the network formed by inverting the direction of every edge. If we then assign a weight of 1 to every edge, a threshold $\theta = 1$ to the network, and the only possible $DNF(.) = V_1(.)$, we recover the same pretopological space. In other words, we are adding to the pseudoclosure of A every element that has at least one connection to a member of A.

This is a good example of the precautions one needs to take about the direction of the edges. If one were to translate the procedure into natural language, one would loosely say that when working with the multivalued function, one takes a set and calculates the pseudoclosure by seeing how far it spreads. Our framework, on the other hand, always takes the perspective of an external element and asks how strongly is its connection to the set. By changing the order of the edges we are basically saying the connection of an element to a group is strong if the group can spread to it.

3.2.1.5 Pretopology and binary relationships

Let us consider a family $(R_i), i = 1, \dots, n$ of binary relationships on a finite set U and let us define $\forall i = 1, \dots, n,\ \forall x \in U,\ R_i(x) = \{y \in X\}$. Then, the pseudoclosure:

$$a_s(A) = \{x \in U \mid \forall i = 1, \dots, n, R_i(x) \cap A \neq \emptyset\}$$

is called the strong pretopology.

It is easy to see that this fits into our framework since this is exactly what we have been doing for all previous examples, to define binary relations from the prefilter basis set. Here we already have the networks defined by the relations, the weights and the thresholds are again all equal to equal to 1, and the $DNF(.)$ is again the conjunction of all the boolean functions associated to each network.

In the same manner, we can define:

$$a_w(A) = \{x \in U | \ \exists i = 1, \dots, n,\ R_i(x) \cap A \neq \emptyset\}$$

which is called the weak pretopology.

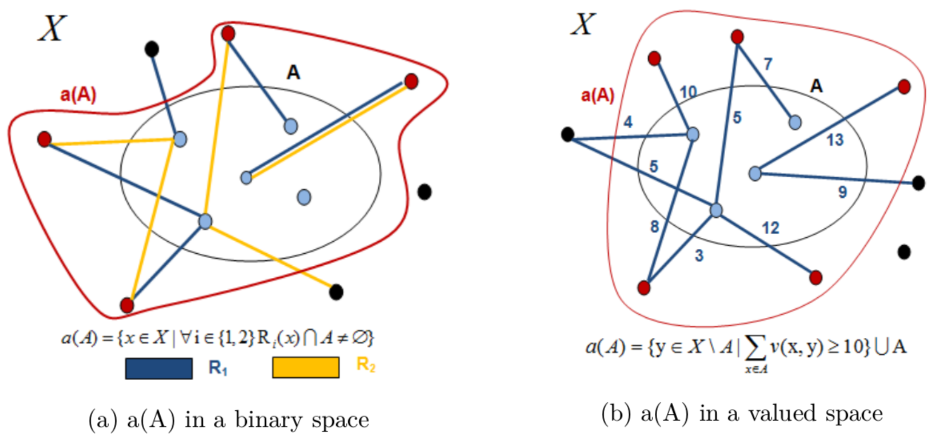
We finish this subsection by presenting two propositions:

Proposition 3.1. *$U, a_s(.)$ is a pretopological space of type V ([?])*

Proposition 3.2. *$U, a_w(.)$ is a pretopological space of type V_D ([?])*

The last proposition might be particularly interesting, since as we already mentioned, a V_D space can be associated to a single graph. Indeed, if we ask for at least one connection from an element x to the set A in order for x to belong to the pseudoclosure, we can put all relations into one single network and study the pseudoclosure equivalently.

3.2.1.6 Pretopology and valued relationships


X
a(A)
A
$a(A) = \{x \in X \mid \forall i \in \{1,2\} R_i(x) \cap A \neq \emptyset\}$
R₁
R₂
X
a(A)
A
10
7
4
5
13
5
9
8
12
3
$a(A) = \{y \in X \setminus A \mid \sum_{x \in A} v(x,y) \geq 10\} \cup A$

(a) a(A) in a binary space (b) a(A) in a valued space

Figure 3.7: Examples of pseudoclosure

The last practical way to build a pretopology identified in [29] is that of a pretopology built over a set of weighted relationships. This is introduced by defining a real valued function $\nu : U \times U \longrightarrow \mathbb{R}$ that takes each relation to a real number. The pseudo-closure $a(.)$ is then defined in the following way:

$$\forall A \in \wp(U), a(A) = \{y \in U - A \mid \sum_{x \in A} \nu(x, y) \geq s\} \cup A; s \in \mathbb{R}$$

This is exactly the definition of our framework for the case of one graph G_1 whose weights are the values of the relations, $\Theta = \{\theta_1\} = \{s\}$ and $DNF(.) = V_1(.)$.

As we have shown, all previous situations can be seen as particular cases of the neighborhood definition, where each element has a family of neighborhoods associated to it, and we ask that either all -for the *strong* pretopology- or at least one of those neighborhoods -for the *weak* pretopology- intersects the set. This is the first time where there is no obvious way to conceive the problem in that manner. The best way to understand how much general this is, it is by remembering that the *strong* and the *weak* pretopologies are always of $V-type$. Now, it is easy to build an example of pretopology over a valued relationship where that is not the case.

Let us take a graph with three nodes x, y and z, two edges (x, y) and (x, z), and relationship values $\nu((x, y)) = 2$ and $\nu((x, z)) = -2$. If we build a pretopology with $s = 1$ then it is easy to see that $x \in a(\{y\})$ but $x \notin a(\{y, z\})$. So we have a situation where $A \subset B \wedge a(A) \not\subset a(B)$, in other words, we have pretopological space that's not even a $V - type$.

Non $V - type$ spaces have been largely ignored in the literature, because their generality doesn't allow them to have many properties. In particular, they cannot assure the existence of the closure of a set, and the structural algorithms that we will see in the next section rely on that assumption. We find this regretful, since as we mentioned in the introduction, many real life situations can be naturally modeled with a non $V-type$ space. We try to overcome this difficulty in the last section of the Structure chapter.

3.2.2 Other examples found in the literature

3.2.2.1 Co-authorship pretopology

Dalud-Vincent [44] defines two pretopological spaces to characterize the structure of co-authorship over a set of academic paper authors. In the first one, an author belongs to the pseudoclosure of a set, if the set contains all other co-authors of at least one paper the author has published. The second pretopology is more restrictive; an author will belong to the pseudoclosure of a set, if the set contains all other co-authors of a paper where he/she was the first author. These are two good examples of structures where some information would be lost if we tried to characterize them with a single network.

Although these two pretopologies seem extremely specific at the first sight, they can still be modeled in our framework in the following way:

- In the first case, we can define associate a network to each paper. Connections will exist between all the authors of the paper and the rest will be isolated nodes. The weights of each link will be equal to one, and the threshold for each network will be equal to the number of authors minus one. The $DNF(.)$ in this case will be $V_1 \vee V_2 \vee ... \vee V_n$, where n is the number of different networks. It's easy to see that the only way the belonging of an individual to the pseudoclusure of a set will evaluate to **True** in a particular network, is if all of his/her co-authors in the paper represented by that network are part of the set. The $DNF(.)$ evaluates to **True** as soon as one network does, so having the co-authors of just one paper suffices to join the author to the pseudoclosure.

- The second case is similar, except the networks that represent each paper only have links relating the first author to the others.

3.2.2.2 n-relations pretopology

In 1999 [23] Largeron and Bonnevay preseted an structural algorithm that will be discussed in detail in the next section, and used it in a practical case. Although the case treated in detail in this paper is just that of a weak pretopology, where each element has a prefilter basis of three elements, it's interesting to notice that at the very end they mention the possibility of defining a pretopology over a set of binary relations, by asking that at least three relations connect an element to the set. This is something that doesn't fit in the schema of prefilter basis as it was defined at the time, but it seems that the idea of defining complex instructions was already present. Still, as far as we know, a formalization of that same idea didn't come until recently [41].

A generalization of that same idea can be also modeled in our framework. First we need to transform the prefilter basis into networks just as we did before. The $DNF(.)$ will be the disjunction of all conjunctions of three boolean variables V_i.

3.2.2.3 Multilayer valued relationships

In [91] Levorato generalizes the valued relationship pretopology by using two valued relationships at the same time.

Levorato uses the study presented in [115], where a dataset with multiple valued relations of monks in a cloister is presented. He takes two of these relations, *esteem* and *disesteem* , and studies the question "who wants to join my group?". For this he defines a pseudoclosure under the assumption that people with great esteem for a team will try to join it, but they won't be accepted if the desesteem the group feels for them is high. This is formalized by imposing a minimum threshold of esteem from x to A to join, and a maximum threshold of desesteem from A to x accept. An example can be seen in figure 3.8.

Although at first sight this might seem like a particular case of our framework with two networks, one should be careful with the direction of the relations. Indeed one of the criteria of the pseudoclosures concerns relations going into the group, while the other concerns relations going out of the group. This of course can be easily fixed, as we saw in the section about neighbor functions. We just need to invert the sens of the arrows. We are basically changing from a graph that represents x *connects to* y to one that represents y *is connected by* x.

Since both relations are quantified with positive values, the space generated is of type V, and the closure of each subset is guaranteed. He author uses this and calculates then the elementary closed sets (*i.e.* the closures of each singleton) in order to study the coalitions than are formed in this manner.

As with the example mentioned in the previous subsection, although the author is aware and makes use of the power of pretopology as a tool to deal with multi-criteria rules, the idea of formalizing this into a unique and concrete framework is not established.

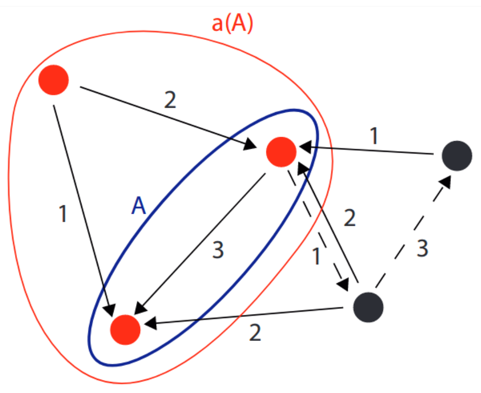


Figure 3.8: Pseudoclosure of A with $\alpha = 3$, $\alpha = 1$. Esteem relation is plain and disesteem relation is dashed.

3.2.2.4 LSP

Recent work has been done on the concept of learning a pretopology. We mention it here for the sake of completeness but it will be treated in full detail in the last chapter of this part of the thesis.

3.2.3 General pretopology

A similar strategy to the one used for the co-authors example can be used to generate any possible V-type pretopology from a partial list of its pseudoclosures. For each set A and its pseudoclosure $a(A)$ we create a network where every element that belongs to $a(A) - A$ is connected to every element of A. We give weights equal to 1 to all the links, and we associate a threshold equivalent to the number of elements on A. The $DNF(.)$ will be again $V_1 \vee V_2 \vee ... \vee V_n$. Let's call $a_F(.)$ the pseudoclosure obtained by applying the strategy just described.

It's obvious by construction that for any tuple $(A, a(A))$ received as input, $x \in a(A) \implies x \in a_F(A)$. Now, let's suppose that we have some elements that belong to the estimated pseudoclosure, without being in the original one:

$$\begin{aligned} & x \in a_F(A) \wedge x \notin a(A) \\ \iff & \exists B \subset A \mid x \in a(B) \wedge x \notin a(A) \\ \iff & \exists B \subset A \mid a(B) \not\subset a(A) \end{aligned}$$

So as long as the list of pseudoclosures received as input is consitent with a V type space, then for every A received as input, the estimated pseudoclosure $a_F(A)$ will be equal to $a(A)$.

3.3 Possible generalizations

We will end this chapter presenting some possible generalizations of pretopology that we can obtain by relaxing some conditions on the framework:

- By replacing the networks that constitute the pretopology by random networks, we recover the notion of stochastic pretopology as presented in [30], while keeping the practical advantages of the framework.
- By replacing the networks that constitute the pretopology by dynamic networks, we get the notion of a dynamic pretopology. This notion, as far as we know, has never been used. It's important not to confuse this notion of a dynamic pretopology, with the fact that a normal (*i.e.* static) pretopology can be naturally used to study dynamic phenomena such as diffusions or transformations.
- Identifying the minimum threshold for an element to be part of the pseudoclosure of a set could be interpreted as a weight. The notion of a weighted pretopology is also one that has never been used, but it seems like a natural way to quantify the strength of the link between an element and a group.

Chapter 4

Pretopological Metrics

In the context chapter we introduced most of the classical metrics that have been defined on a pretopological space, and we saw that most of those that applicate to a set can be seen as a transformation of the pseudoclosure function. A good knowledge of the image of this function over the space is then fundamental to understand the characteristics of that space. In this part we focus on the particular problem of finding the biggest pseudoclosure for sets of a fixed size. Having a big pseudoclosure, as having a big degree in the case of a graph, can be seen as a measure of centrality, and for most practical applications as a measure of importance. All of the results that are presented here can be trivially adapted to study the opposite problem of the smallest pseudoclosure.

The first part shows that the biggest pseudoclosure can be found using Integer Linear Programming techniques, for certain pretopological spaces.

The second part introduces a notion that formalizes the idea of "someone that attracts people to a group when he/she makes part of it", and presents an efficient way of calculating it for certain spaces. This notion was used to find sets with a big pseudoclosure, which we proved to be a good way of measuring centrality in [81]. Those results are presented in section 8.2.

4.1 ILP Biggest Pseudoclosure

One advantage of the framework presented is that by working with the matrix representation of the networks that define the pretopology, we can see the calculation of the pseudoclosure as a system of linear equations for some particular cases. This will allow us to treat the problem of finding the biggest pseudoclosure as an integer linear programming (ILP [118]) problem. To our knowledge, Pretopology has never been studied with the tools of ILP before.

Let's begin with a pretopology built over a singular graph with n nodes, weighted adjacency matrix WA and threshold θ, and let's define:

$$\begin{aligned}
\mathbf{c} &= \left[\begin{array}{ccc|ccc} 0 & \dots & 0_n & 1_{n-1} & \dots & 1_{2\cdot n} \end{array}\right] \\
\mathbf{b} &= \theta_{2n\times 1} \\
\mathbf{d} &= \left[\begin{array}{ccc|ccc} 1 & \dots & 1_n & 0_{n+1} & \dots & 0_{2\cdot n} \end{array}\right] \\
\mathbf{s} &= \left[\begin{array}{c} s \end{array}\right]
\end{aligned}$$

$$A = \left[\begin{array}{ccc|cccc} wa_{11} & \dots & wa_{1n} & \sum_{wa_{1i}<0} \mid (wa_{1i}) \mid +\theta & 0 & \dots & 0 \\ wa_{21} & \dots & wa_{2n} & 0 & \sum_{wa_{2i}<0} \mid (wa_{2i}) \mid +\theta & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ wa_{n1} & \dots & wa_{nn} & 0 & 0 & \dots & \sum_{wa_{ni}<0} |(wa_{ni})| + \theta \end{array}\right]$$

We'll show now that solving the following ILP problem gives us the set with s elements that has the biggest pseudoclosure on the space:

$$\begin{array}{lrl} \text{minimize} & \mathbf{c}^T \cdot \mathbf{x} & \\ \text{subject to} & A \cdot \mathbf{x} & \geq b \\ & d \cdot \mathbf{x} & = \mathbf{s} \\ x_j \in \{0,1\}, & j = 1, ..., 2n & \end{array}$$

Let's analyze the equation in order to better understand why this is the case.

We are looking for a target set ts, whose membership vector will be denoted $\mathbf{ts}$, that will have the biggest pseudoclosure among all sets with s elements. The vector $\mathbf{x}$ will end up being the concatenation of $\mathbf{ts}$ and $\mathbf{a(ts)^C}$, the membership vector of the complement of the pseudoclosure of ts. In other words, the first half of $\mathbf{x}$ will have zeros when an element doesn't belong to ts, and will have 1's when they do. The second half will have 0's if an element belongs to the pseudoclosure of ts and 1's otherwise.

If we just multiplied the weighted adjacency matrix $\mathbf{WA}$ (*i.e.* the left half of the matrix $\mathbf{A}$) by $\mathbf{ts}$, then $(\mathbf{WA} \cdot \mathbf{ts})_i$, the ith component of the result, would represent the weighted sum of the edges going from element i to the nodes in ts. Now, this sum is exactly what needs to be bigger than the threshold θ in our framework, in order for the element to belong to the pseudoclosure of ts.

Let call $\mathbf{D}$ the diagonal matrix constituted by the right half of A. Each element $D_{i,i}$ is designed so it can compensate any effect that the negative edges going from an element i to a set could have in the weighted sum. In other words, no matter which set ts we select, as long as $\mathbf{x}_{n+i} = 1$, $(A \cdot x)_i \geq \theta$.

Now let's go back to the ILP problem. A trivial feasible solution to the problem would be to take any vector $\mathbf{ts}$ with s 1's and set the whole second half of the vector $\mathbf{x}$ to $\mathbf{1}$.

Since we are trying to minimize the sum of that second half, then we would like to change the biggest possible number of 1's into 0's, but the only way to change $\mathbf{x}_{n+i} = 1$ to $\mathbf{x}_{n+i} = 0$ and still have a solution is if $\sum_{j=1}^{n} wa_{ij} ts_i \geq \theta$. Simply put, we can change to 0 the components representing the elements that belong to the pseudoclosure of ts.

In conclusion, the more 0's we have in the second half of a solution vector x the biggest the pseudoclosure will be, since those 0's indicate the elements that belong to the pseudoclosure. Minimizing over all possible vectors will get us the set with the biggest pseudoclosure of all.

Finally, we notice that it's equation $\mathbf{d} \cdot \mathbf{x} = \mathbf{s}$ that guarantees us that ts will have s elements.

Now that we see how the solution to that ILP problem gives us the biggest pseudoclosure for a set of size s, let's see two ways of extending this to answer some other questions.

First, let's suppose we want to exclude a set e of size s, represented by the vector $\mathbf{e}$, from the search space of solutions. It would be enough to add the following inequation in order to rule the set out:

$$\left[\begin{array}{c|ccc} \mathbf{e} & 0_{n+1} & \dots & 0_{2 \cdot n} \end{array}\right] < [s]$$

Indeed, we would be forcing the solution to include less than s elements from set e, and to have exactly s different elements, so the solution cannot be a vector $\mathbf{x}$ whose first half equals $\mathbf{e}$.

Doing this iteratively will allow us to find the n sets with the biggest pseudoclosure. We can begin by finding the first one and then removing it from the options in the next iteration. We then do the same and we'll find the second biggest pseudoclosure. We repeat the procedure n times.

The second extension is the one that allows us to apply this procedure to a pretopology defined by a single conjunction $DNF = V_1 \wedge \ldots V_n$. We just need to make the following substitution:

$$A = \begin{bmatrix} A_1 \\ \hline \vdots \\ \hline A_n \end{bmatrix}$$

Where $A_i = [WA_i, D_i]$ is the matrix corresponding to the graph G_i of the pretopology.

Integer Linear Programming is known to be NP-hard, so attacking the problem of the biggest pseudoclosure this way won't guarantee us a good performance in every case. On the other hand, let us suppose we have a polynomial time algorithm to find the biggest pseudoclosure over sets of size s. Then let us take the most basic pretopology, built over a single unweighted and undirected graph. The greedy algorithm that starts finding the biggest pseudoclosure over all singletons, then over all sets of size two, and continues until it finds a set whose pseudoclosure is the whole set, would still be polynomial. Now, solving that is equivalent to solving the minimum dominating set problem [78, 127], a problem that is also well known to be NP-hard.

Since we now know that we are dealing with an NP-hard problem, the importance of connecting it to ILP becomes clear. ILP is a very active research field, so it's interesting to be able to profit from the multitude of heuristics that are constantly developed to solve such problems.

4.2 Team Builder Concept

When using pretopology to model a system or a diffusion inside of a system, we are normally interested in groups that have a large pseudoclosure. We propose here to concentrate on the individuals that on average will increase the most the pseudoclosure of a group.

We start by defining the function group teambuilder index (gti) as:

$$gti(x, A) = \mid a(A \cup x) \mid - \mid a(A) \mid$$

Where $\mid A \mid$ is the cardinality of the set A.

In other words, we calculate the pseudoclosure of a set A not containing an element x, and then we calculate the pseudoclosure of the union of A with x, if the difference between the two is large, then the element x made a big contribution. By doing this over all possible sets, we are able to identify the members that on average contribute the most to enlarging a set.

That is to say we are looking for the elements that have the largest teambuilder index, which is defined as follows:

$$teambuilder(x) = \sum_{A \subseteq U \mid x \notin A} gti(x, A) = \sum_{A \subseteq U \mid x \notin A} \mid a(A \cup x) \mid - \mid a(A) \mid$$

A naive approach to this calculation would involve a sum of as many elements as subsets on the set U, which becomes quickly intractable. While this might be the only way to calculate the index for any possible pretopology, we will concentrate here in the particular case of the pretopology defined over one single graph, with all weights and the threshold equal to one, and the only possile $DNF(.) = V_1$

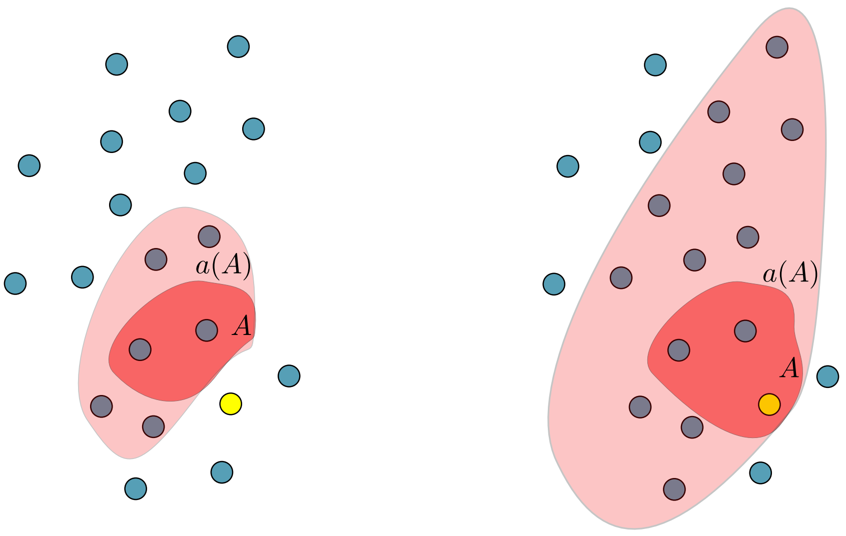


Figure 4.1: Example of the Teambuilder concept

Here the value of the teambuilder is equal to 5.

Under these circumstances the *teambuilder* index is given by the following formula:

$$\sum_{y \in neighbors(x)} 2^{n - |neighbors(y)| - 1}$$

To see why this is so, we need to change from asking "for each set A such that $x \notin A$, how many elements would x add to the pseudoclosure?", to ask "for each element z, how many sets A such that $x \notin A \land z \notin a(A)$ will have z in their pseudoclosure after joining x ($i.e.$ $z \in a(A \cup x)$)".

Adding x will only add elements that are connected to x in the network, so those are the only z's that need to be considered. And those elements z will be added to the pseudoclosure of all those sets of that don't contain z in their pseudoclosure.

So the answer to the question is simply "all those sets that don't have z in their pseudoclosure already", and this for z connected to x. The way to calculate how many those sets are, is by taking one z at a time, and calculate all those sets that don't include it in the pseudoclosure. Since to include z in its pseudoclosure, a set A should contain one of its neighbors in the network, we just need to consider all the subsets of the $n -|neigbors(z) - 1|$ elements that do not include any neighbor, nor z.

In practice we won't need to calculate those powers of 2, since we will only be interested in comparing the difference of teambuilder index between elements, and this can be done by just stocking the different exponents of 2 that we are adding. So the $teambuilder$ index can be calculated for all x in $O(E)$, where E is the set of edges of the network.

Although it is a fundamentally pretopological measure, the $teambuilder$ index is associated to each node, and for the case of a pretopology built according to a network it can be easily translated into a network metric. It is, to our knowledge, the only pretopological metric that is proper to an element.

In figure 4.2 we can appreciate how the $teambuilder$ index is a sort of compromise between degree and betweenness centrality. Indeed, the highest $teambuilder$ is not necessarily the person with more contacts, since for a very clustered network, the addition of that node to a group would not add many new neighbors if the group was already connected to most of its contacts.

The $teambuilder$ is a way of quantify the importance of an element as a mean of attracting other elements to a coalition. We will explore in the application part if this notion can be considered as a good measurement of node centrality.

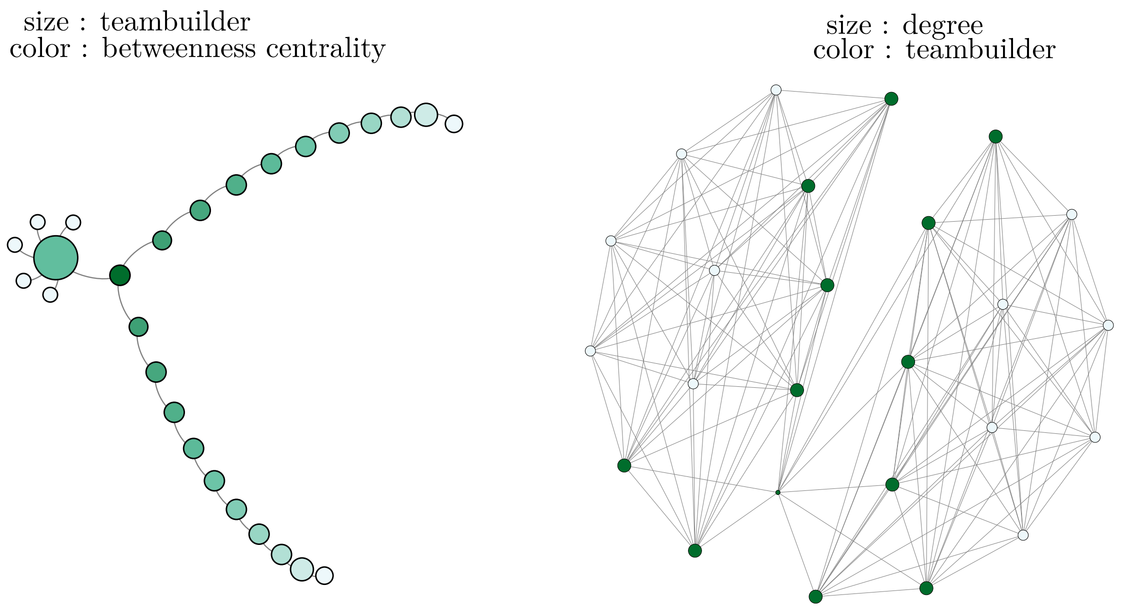


Figure 4.2: Teambuilder index versus degree and betweeness centrality

On the left we see the difference between the *teambuilder* index and the degree of a node. On the right we see the difference with the betweenness centrality.

Chapter 5

Pretopological Structures

Once we have a pretopological space structuring a set, we can do some further structural analysis by grouping elements that are homogeneous under some characteristics of the space. This could be assimilated to the notion of finding the biggest connected components in a graph.

Two algorithms have been presented to analyze the structure of a pretopological space. They both work on pretopological spaces of type V since they need to be sure of the existence of the closure of a set. We will study both of them, presenting some improvements.

We will end the chapter presenting a generalization of the first algorithm that will allow us to work with any kind of pretopological space. We'll see in the last part of the thesis that this generalization has interesting applications in the context of clusterization.

5.1 Structural Analysis

In [83, 23, 84] Largeron and Bonnevay introduced an algorithm that makes a structural analysis of a pretopological space. The goal of this algorithm is to highlight groups of "interdependent" elements.

Before presenting the algorithm some definitions are necessary:

Definition 5.1. *An elementary closed set is the closure of a singleton $\{x\}$ of U. We will denote it F_x.*

Definition 5.2. *A closed set is said to be Minimal, if no other closed set is contained in it.*

We will denote F_e the family $\{F_x \mid x \in U\}$ of all elementary closed sets of U.

The algorithm is designed to be used in pretopological spaces of type V, since the following facts are used by it:

- In a $V-type$ pretopological space every set has a closure.
- In a $V-type$ pretopological space every minimal closed subset is necessarily in F_e.
- In a $V-type$ pretopological space two distinct elementary closed subsets F_x and F_y are either disjoint ($F_x \cap F_y = \varnothing$) or contain a nonempty intersection such that $\forall z \in F_x \cap F_y$, we have $F_z \subset F_x \cap F_y$. The proof of this can be found in [**?**]

The algorithm is structured in three phases: first we calculate the family of elementary sets; then we explore this family selecting the minimal subsets; finally we establish

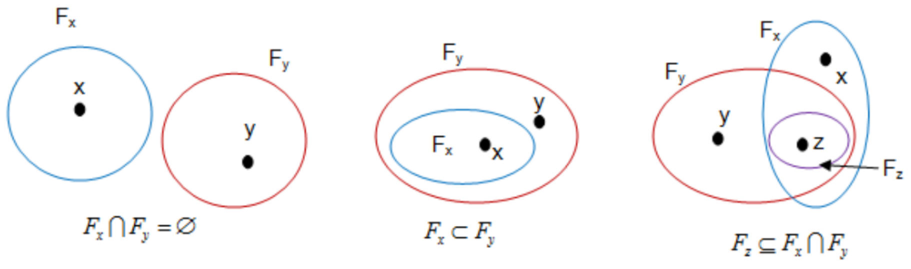


Figure 5.1: Intersection of closures

a structural relation among all subsets. The pseudocode of the whole process, as it was presented in the paper, can be seen in algorithm 5.1

The algorithm is presented as a way to study which elements could potentially be reached by others, and which elements would these other be. The interpretation given to the notion of reaching depends of course on the model maker. The notion has actually proven to be quite flexible, having been used recently as a way to extract taxonomy hierarchies on a conceptual corpus [41]. The pseudoclosure defined in the paper is such that a word can expand to reach concepts that are more general than it. It has also been used in the context of clusterization, as a way to infer some of the parameters of the $K-means$ algorithm that are usually set up manually by the researcher [87, 88, 32].

Algorithm 5.1 Structural Analysis

```
Require: A pretopological space $(U, a(.))$
1: procedure StructuralAnalysis($(U, a())$)
2:     $F_e \leftarrow$ ElementaryClosedSubsets($(U, a)$)
3:     $F_m \leftarrow$ MinimalClosedSubset($F_e$)
4:     $A_s \leftarrow$ ExtractStructure($F_e, F_m$)
5: end procedure
```

Before going into the details of the modifications to the **ElementaryClosedSubsets**() function that we will make, we mention that also the **ExtractStructure**() procedure needs to be slightly changed, since there's no output in the way it was presented. The relation "G is a descendant of F" is simply stated every time two sets having it are found in the process of scanning all elementary closed sets. Now, it seems that the most natural output of the procedure would be an acyclic directed graph. In the graph there will be an arrow from a set A to a set B if and only if $A \subset B$ and $\nexists C \mid A \subset C \wedge C \subset B$. An example of the hierarchy extracted through the algorithm can be found in figure 5.5, in the section presenting the quasihierarchy concept.

Going back to the **ElementaryClosedSubsets**() procedure, as it can be seen in the pseudocode, when we calculate the closure of an element we apply the pseudoclosure repeatedly until it stops expanding. Now, this doesn't take into account that if two or more singletons converge to the same set at some point, there's no need to calculate the closure of that set every time. Since calculate the pseudoclosure is a very expensive operation (more details at the end of the section), we present a modification of the algorithm that tries to correct this.

The basic idea of the algorithm is to make the elementary sets grow more or less in parallel, by applying always the pseudoclosure function to the smaller sets we have calculated so far. By doing this, and keeping track of the different sets we get through that process, we will need to calculate each pseudoclosure only once. The pseudocode

```
procedure ElementaryClosedSubsets($(U, a())$)
    $F_e \leftarrow \varnothing$
    for all $x \in U$ do
        $F \leftarrow a(\{x\})$
        while $a(F) \neq F$ do
            $F \leftarrow a(F)$
        end while
        $F_e = F_e \cup F$
    end for
    return $F_e$
end procedure

procedure MinimalClosedSubsets($(F_e)$)
    $F_m \leftarrow \varnothing$
    while $F_e \neq \varnothing$ do
        $F \leftarrow F_e.pop()$        ▷ We remove an element from $S_e$ and assign it to F
        $minimal = True$
        $F_t = F_e$
        while $(F_t \neq \varnothing) and minimal$ do
            $G \leftarrow F_t.pop()$
            if $G subset F$ then
                $minimal = False$
            else if $F \subset G$ then
                $F_e.remove(G)$
            end if
        end while
        if $minimal$ then
            $F_m.add(F)$
        end if
    end while
    return $F_m$
end procedure

procedure ExtractStructure($(S_e, S_m)$)
    $Q \leftarrow \varnothing$
    for all $F \in S_m$ do
        $Q.enqueue(F))$
    end for
    while $Q \neq \varnothing$ do
        $F \leftarrow Q.dequeue()$
        $S \leftarrow \{G \in S_e, F \subset and F \neq G\}$        ▷ Supersets of $F$
        for all $G \in$ MinimalClosedSubsets($S$) do
            if $G \notin Q$ then
                $Q.enqueue(G))$
            end if
            G is a descendant of F
        end for
    end while
end procedure
```

of the new version can be found in 5.2.

Before going any further in the complexity analysis, we attract the attention to the input of the function. We present the pseudocode of our version taking a list of sets as input, since we will use this more general version later. Obviously, to get it to calculate the elementary closed sets we just need to pass it the list of all singletons.

By looking at the pseudocode one could think that our algorithm has the disadvantage of losing track of the closure of each particular element, but we know from the properties of V spaces that the closure of each element will be the smallest elementary closed set that will contain it.

Another advantage of the modified **ElementaryClosedSubsets**() is that it gives us the list of closures with unique elements, so the searching space for the rest of the algorithm could be much smaller.

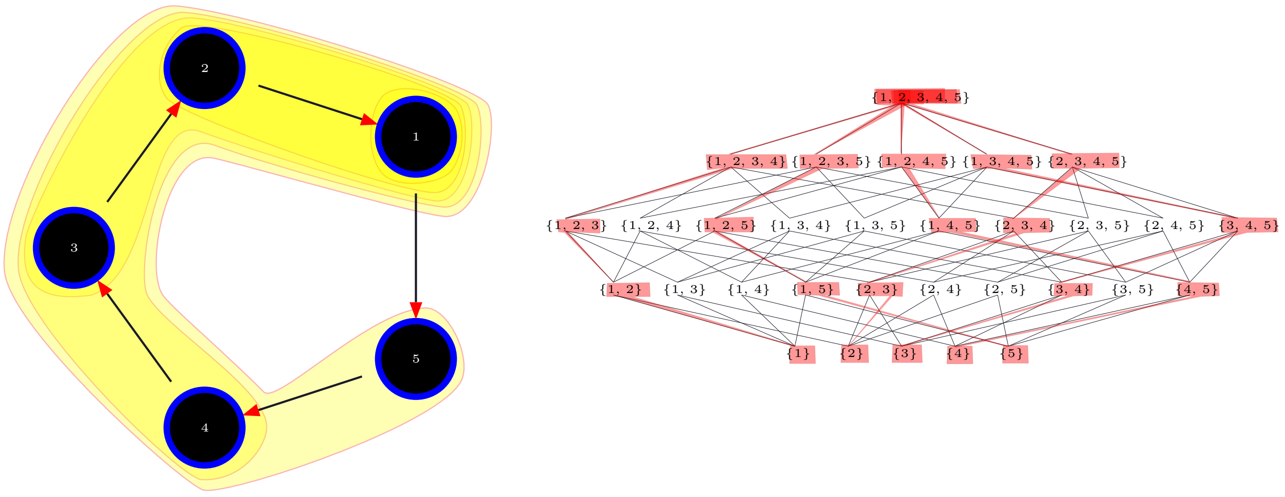


Figure 5.2: Structural Analysis worst case scenario

A type of pretopology that will never have two different subsets with the same pseudoclosure.

Algorithm 5.2 ElementaryClosedSubsets new version

```
 1: procedure ElementaryClosedSubsetsNV($(U, a(.)), seedsList$)
 2:     $F_e \leftarrow [list()\ for\ i\ in\ Size(U)]$          ▷ An array of lists
 3:     for all $seed \in seedsList$ do
 4:         $F_e[Size(seed)].append(seed)$
 5:     end for
 6:     for all $i = 1 \rightarrow Size(U)$ do
 7:         for all $set \in F_e[i]$ do
 8:             $pseudoclosure \leftarrow$ Pseudoclosure($set$)
 9:             if $pseudoclosure \notin F_e[Size(pseudoclosure)]$ then
10:                 $F_e[Size(pseudoclosure)].append(pseudoclosure)$
11:             end if
12:         end for
13:     end for
14:     return $F_e$
15: end procedure
```

If we only consider the number of pseudoclosures calculated as the unit measure of complexity, then our algorithm would have the same worst case complexity than the original version. This is n^2, where n is the number of elements in U. That is the case when each application of the pseudoclosure adds only one element at a time, they all have the space U as their closure, and no two elements converge to the same set before reaching their closure. A simple example of this situation can be seen in figure 5.2. Circular graphs of that form, in a pretopology built over that single graph, all weights and the threshold equal to 1, and $DNF(.) = V_1(.)$, will grow in a manner that the number of pseudoclosures necessaries to get all elementary closed sets will always be equal to n^2.

On the other hand, we could have cases where the number of pseudoclosures calculated varies greatly between the two algorithms. Such situation occurs when every singleton converges extremely quickly to some set that's part of the growing path of many other elements. We see a simple example of this situation in figure 5.3, where the graphs for the cases $n = 5$, $n = 6$ and $n = 7$ are shown. The parameters of the pretopology are the same as before, all weights and the threshold equal to 1, and $DNF(.) = V_1(.)$.

To better understand the difference we can look at the Hasse diagram with the pseudoclosures for case $n = 5$ in 5.4. There, we can clearly appreciate that for a graph with n elements, the number of pseudoclosures calculated for the original algorithm will be:

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$$

While the number of pseudoclosures calculated by the new version is just:

$$\sum_{i=1}^{n-1} 1 + \sum_{i=1}^{n-1} 1 = 2(n-1)$$

To get the closure of the first singleton we would need to apply $(n-1)$ pseudoclosures, while all other elements will just need one application. We pass then from a quadratic number of calculations to a linear one.

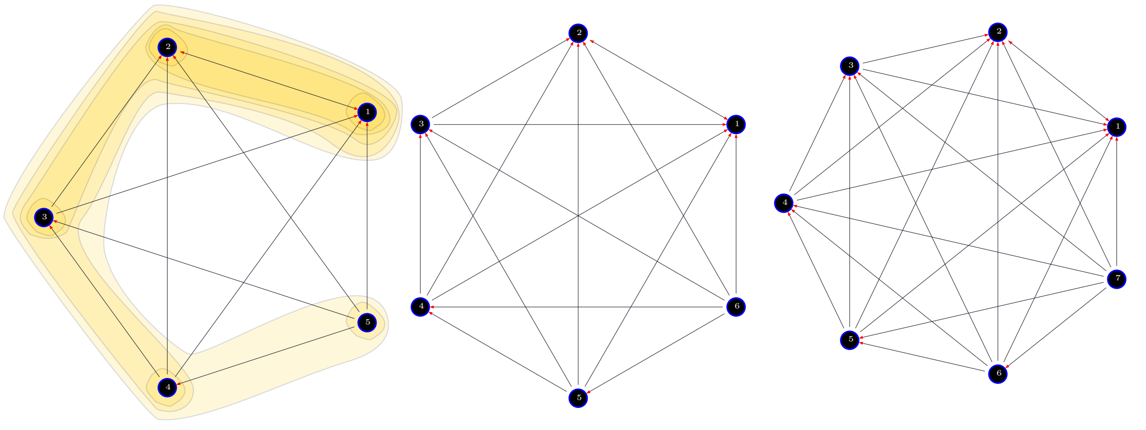


Figure 5.3: Pretopologies with $2(n-1)$ versus $\frac{n(n-1)}{2}$ pseudoclosure calculations

Here we see the first members of a family of pretopologies where the number of pseudoclosure calculations for our algorithm is just $2(n-1)$, while it is $\frac{n(n-1)}{2}$ for the original version.

Until here we have been focusing only on the number of pseudoclosures calculated, but the situation is more complicated than that. To make things more concrete we will briefly describe how things are implemented in the library that will be presented in chapter 7:

- Sets are represented by the same kind of membership arrays of size n described in the ILP section, where n is the number of elements in U.

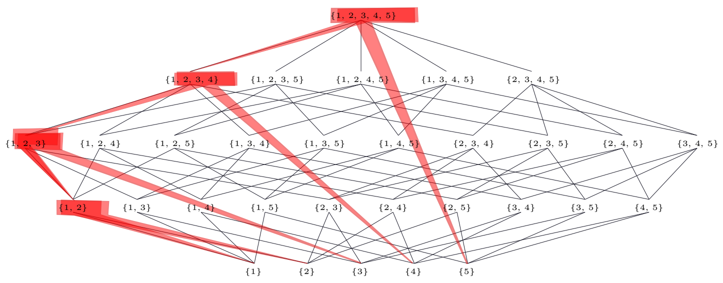


Figure 5.4: Hasse diagram of $2(n-1)$ versus $\frac{n(n-1)}{2}$ pseudoclosure calculations

Hasse diagram showing the pseudoclosures needed to get the closures of each singleton. Going from left to right, each singleton will need one pseudoclosure calculation less than it's predecessor.

- The pseudoclosure takes $q(n^2+n)+C$ instructions, where q is the number of different networks and C is the constant time needed to evaluate the $DNF(.)$. Indeed, to see what's the pseuclosure $a_i(.)$ of a set represented by the vector $\mathbf{x}$ on network i, we just need to multiply $\mathbf{WA}_i \cdot \mathbf{x}$ -where $\mathbf{WA}_i$ is the weighted matrix of network i-, and see which elements of the result are bigger than the threshold. This has to be done for every network before the $DNF(.)$ is evaluated.

- The pseudoclosures are stocked in a python set, a data structure based on a hashtable, that allow us to know if an element is already present in amortized $O(1)$ (although it has complexity $O(n)$ in the worst case).

- We need to *stringify* the array back and forth every time we want to stock it in a set, or calculate its pseudoclosure, in order to make it hashable.

So every pseudocloure calculation in our algorithm will take the usual amount of time, plus the amount of time necessary to *stringify* it back and forth, which is $O(n)$, to calculate the hash and to check if it belongs to the set, which is amortized constant. Although this doesn't alter the order of complexity of the pseudoclosure calculation, it certainly adds some extra time.

In general, the advantage of the new algorithm is strictly dependent on the number of times that a same pseudoclosure is calculated. That, in its turn, depends on a combination of two factors: the number of times the pseudoclosure function has to be applied before stop expanding, and the number of times two or more elements converge to the same set. Indeed, if many elements converge to a set at some point, but that set stop expanding quite quickly afterwards, we won't have much gain.

The two examples presented before are extreme cases. To see how the trade-off between a more expensive pseudoclosure and potentially less calculations works in real life, we tested both algorithms in the six datasets used in the last chapter of the thesis. The algorithm doesn't actually calculate elementary closed subsets, but *elementary closed subsets of degree 4*, which a concept that will be soon introduced. The details of this will be seen in the corresponding chapter, what's important is just the transaction between calculating the pseudoclosure of a set every time it appears, versus stocking it and recover it. The results of this experiments are seen in 5.1, where we can see that the new method was always about three times faster.

On a more general note, those datasets were geometrical points in the plane, and it seems quite natural that elements that are close would rapidly attain a same set in the iterative process of applying the pseudoclosure.

dataset	old version		new version	
	mean	std	mean	std
noisy circles	80.102	1.996	23.443	0.268
noisy moons	56.064	1.059	12.963	0.152
varied blobs	22.193	0.361	9.106	0.129
anisotronic	8.685	0.076	3.512	0.057
blobs	7.188	0.156	2.855	0.059
no structure	91.562	1.775	36.737	0.547

Table 5.1: Time performance (in seconds) of both algorithms over 20 runs, with the 6 datasets of chapter 9

5.2 Structural Equivalence

In [47] an algorithm was presented to find the structure of social groups when we have a network describing social relations. The paper proposed a new way of defining structural equivalence inside of a network by iteratively extracting nodes that were at the periphery of the network, *i.e.* nodes that became isolated by the removal of a single other node. The algorithm aggregated at each iteration the nodes at the periphery in one group, and those that when removed left others isolated in another group.

After structuring the space, the authors compared the groups found by the algorithm with other characteristics of the population and found that socio-demographical differences corresponded to network differences. The pseudocode of this algorithm can be seen in algorithm 5.3

The paper ended up pointing out that the algorithm could be adapted to pretopologies, but without going into the details. Very recently [44] the pretopological version was published with a pseudocode that's the same as before, but where a few concepts should now be interpreted as the pretopological counterparts of the originals.

The definitions necessary to understand this new version are the following:

Definition 5.3. *We say that a space $(U, a(.))$ of type V is strongly connected $\iff \forall C \subset U, C \neq \emptyset, F(C) = U$.*

Definition 5.4. *We say that a space $(U, a(.))$ of type V is connected $\iff \forall C \subset U, C \neq \emptyset, F(C) = U \vee F(X - F(C)) \cap F(C) \neq \emptyset$.*

Definition 5.5. *For any $A \subset U$ in a pretopological space $(U, a(.)$ of type V, we define the pseudoclosure induced by $a(.)$ over A, and denoted $a_A(.)$, as: $\forall C \in \wp(A)$, $a_A(C) = a(C) \cap A$. $(A, a_A(.))$ is called a pretopological subspace of $(U, a(.))$*

Definition 5.6. *$(A, a_A(.))$ is a connected (resp. strongly connected) pretopological subspace of $(U, a(.))$, if and only if $(A, a_A(.))$ as a pretopological space is connected (resp. strongly connected).*

Definition 5.7. *$(A, a_A(.))$ is a biggest connected (resp. strongly connected) subspace of $(U, a(.))$, if and only if $(A, a_A(.))$ is a connected (resp. strongly connected) subspace of $(U, a(.))$ and $\forall B \in \wp(U)$, $A \subset B \wedge A \neq B$, $(B, a_B(.))$ is not a connected (resp. strongly connected) subspace of $(U, a(.))$.*

Definition 5.8. *$b \in A$ is an articulation point of A in $(U, a(.))$ if $(A - \{b\}, a_{A-\{b\}})$ is not a connected (resp. strongly connected) subspace of $(U, a(.))$*

Definition 5.9. *$b \in A$ is an articulation point of degree 1 of A in $(U, a(.))$, if and only if b is an articulation point, and the smallest of the biggest connected (resp. strongly connected) subspaces of $(A - \{b\}, a_{A-\{b\}})$ has cardinality 1.*

Definition 5.10. *$c \in A$ is a weak point of A in $(U, a(.))$ if and only if $\exists b \in A - \{c\}$ such that b is an articulation point of degree 1 of A in $(U, a(.))$ and $\{c\}$ is a biggest connected (resp. strongly connected) subspace of $(A - \{b\}, a_{A-\{b\}})$.*

So we changed the graph version of connected components, the nodes in the periphery (the *weak points*) and those that cause isolation when removed (the *degree 1 articulation points*) for their pretopological homologues. Besides that, the new algorithm takes as input a pretopological space instead of a graph. The rest of the algorithm remains unchanged.

Now, when we look closely at this new version some problems start to emerge. The first of them was already mentioned in the introduction: the algorithm takes as input a

Algorithm 5.3 Social Groups Structure

```
Require: A graph $G(V, E)$
procedure SocialGroupDecomposition($G(V, E)$)
    split the graph G into $CC$ (resp. $SC$)    ▷ connected (strongly) components
    cluster together all the $CC$ singletons (resp. $SC$ singletons)
    consider the set of $CC$ (resp. $SC$) not singleton
    while the set of $CC$ (resp. $SC$) not singleton is not empty do
        consider one of the $CC$ (resp. $SC$)
        remove this $CC$ (resp. $SC$) from the set of $CC$ (resp. $SC$)
        seek the $AP$, the $1 - AP$ and the $WP$ of the $CC$ (resp. $SC$)
        if the number of $WP > 0$ then
            cluster all these $WP$
        else
            if the number of $1 - AP$ already listed, not clustered and present in this $CC$ (resp. $SC$) $> 0$ then
                cluster all these $AP$
            else
                cluster the remaining points of the $CC$ (resp. $SC$)
            end if
        end if
        consider the sub-graph resulting from the removal of the clustered vertices
        split into $CC$ (resp $SC$)
        if the number of $CC$ (resp. $SC$) singletons $> 0$ then
            cluster the $CC$ (resp. $SC$) singletons
        end if
        include every $CC$ (resp. $SC$) not singleton in the set of $CC$ (resp. $SC$)
    end while
end procedure
```

pretopological space. We saw how impractical this instruction was without any further details, but this could now be compensated with the framework proposed in this work.

Perhaps more importantly, before even begin the structural classification, we need to decompose the space into its biggest connected (or strongly connected) subspaces. Now, this was not a problem in the context of graph theory, since many fast algorithms have been developed to find the components of a graph. As far as we know this is not the case for a pretopology.

It would be interesting to notice first the amount of work that a naive approach to the problem of regular connectivity would involve. We would need to calculate the closure of every subset C of U, and when it doesn't equal U, we would calculate the pseudoclosure of the complement to see if the intersection is non empty. Not only we are again confronted to a number of steps equal to the subsets of a set, but for every one of those steps the expensive process of closure calculation would have to be applied.

The case of strong connectivity is certainly more manageable. If we remind the definition of a minimal closed set, we can see that they are connected subspaces. Now, let's take a minimal closed set M and an element y such that $y \notin M$. We know for its definition that $a(M) = M$, so the set $a(M) \cup \{y\}$ won't be a strongly connected subspace. This show us that there can't be a strongly connected subspace bigger than a minimal closed set.

Since connected (and strongly connected) components are a partition of set U [**?**], we can apply the algorithm presented in the former section to calculated every minimal closed set, take those sets out and study the resulting space again for connectivity. If we repeat this process until no more elements are left, we end up with all the strongly connected subspaces.

We couldn't find any algorithm other than brute force to find the connected subspaces, and the possibility of a fast algorithm to do it looks quite challenging. It should be noticed, for example, that a connected subspace of cardinality n may exist even if no subspaces of cardinalty $n - 1$ are connected. So although there's a simple way to test if a connected subspace A is still connected after adding a new element b (we only need to show that $a(A) = A \cup \vee a(\{b\}) \neq \{b\}$), any attempt to find the biggest connected subspaces by systematically increasing the singletons without breaking connectivity is destined to fail.

5.3 Quasihierarchy

We have mentioned how most of the work done with pretopology has focused on $V-type$ spaces, since they have stronger properties than general pretopologies. In particular, the Largeron-Bonnevay algorithm extracts an acyclic graph giving a structural analysis of the space, using three facts about $V-type$ spaces: every set has a closure; every minimal closed subset is necessarily in F_e; two distinct elementary closed subsets F_x and F_y are either disjoint ($F_x \cap F_y = \emptyset$) or contain a nonempty intersection such that $\forall z \in F_x \cap F_y$, we have $F_z \subset F_x \cap F_y$.

Here, we will try to extend these notions, so we can make a similar analysis in two new cases. Before explaining what these cases are, we will introduce some definitions, all of them to generalize the concept of elementary closed set:

Definition 5.11. *We will denote F_{e-n} the family $\{F_A \mid A \in \wp(U) \wedge |A| = n\}$ of all closures of sets with n elements. We will call it the family of elementary closed sets of degree n.*

Definition 5.12. *We will denote $A_{s-n}(x)$, the* ***shortest set of degree n*** *around x: the set built by taking x; selecting the element y connected to x by the smallest weighted edge; doing the same with y instead of x; and repeating the procedure a number n of times. The pseudocode of this process can be seen in algorithm 5.4.*

Definition 5.13. *We will denote S_{s-n} the family $\{A_{s-n}(x) \mid x \in U\}$ of all closures of* ***shortest sets with degree n****.*

Definition 5.14. *We will denote $A_{r-n}(x)$ the* ***random connected set of degree n*** *around x, the set built by taking x, selecting a random element y connected to x, doing the same with y instead of x, and repeating the procedure a number n of times. The pseudocode of this process can be seen in algorithm 5.4.*

Definition 5.15. *We will denote S_{r-n} the family $\{A_{r_n}(x) \mid x \in U\}$ of all closures of* ***random connected sets with degree n****.*

This excess of definitions may seem overwhelming at first sight, but they are all guided by the same idea: start the structuring process not with elements, but with sets, since that's when we see the power of pretopology in action. The first definition is the most natural generalization of the concept of *elementary closed set*, but it becomes too expensive quite quickly -and most likely redundant-, as n grows. The other definitions are heuristic methods to compensate that problem. They will allow us to begin with small, compact sets (not in a topological sense), and see how they extend.

A final definition will be necessary before we extend the structural analysis process. We remind that a closure is defined as the intersection of all closed sets containing a set. We already saw that in the most general pretopological spaces the existence of the closure is not guaranteed. On the other hand, continuous applications of the pseudoclosure to a set and the subsequent images $a(a(...a(a(A))...))$ are destined to converge, since the pseudoclosure is a monotonically increasing function and the size of the space is finite. This allows us to make the following definition:

Definition 5.16. *We call quasiclosure of a set A, the set $QC(A) = a^k(A) \mid a^{k+1}(A) = a^k(A)$*

This is in practice the way we use to calculate the closure of a set in a $V-type$ space. Although it won't be the closure in the general case, because we can't be sure there's not another closed set C such that $A \subset C \wedge QC(A) \not\subset C$, it still might be of interest, since it shows us the limits of the expansion of A through the pseudoclosure.

Finally, by putting all those notions together we will be able to extend the Largeron-Bonnovay algorithm to the following two situations:

- When the pretopological space doesn't comply with the isotony axiom (*i.e.* $A \subset B \not\Longrightarrow a(A) \subset a(B)$).
- When the structural analysis won't be over elementary closed sets, but over some variation of the $degree - n$ elementary closed sets.

To see what these two situations have in common, let's think about the third property of $V - spaces$ mentioned before and used for the structural analysis: two distinct elementary closed subsets F_x and F_y are either disjoint ($F_x \cap F_y = \emptyset$) or contain a nonempty intersection such that $\forall z \in F_x \cap F_y$, we have $F_z \subset F_x \cap F_y$. So if two sets overlap without one being contained in the other, we know there will be a smaller set contained in both, and the graph with the final structure will connect each of those two sets to the smaller one. Now, this won't necessary be the case for any of the two situations presented: in the case of $non - V$ spaces, nothing guarantees that an element in the intersection won't grow beyond it; in the case of elementary $degree - n$ sets, none of the starting sets might be completely contained in the intersection. This makes that no obvious structure may emerge from the collection of quasiclosures.

To overcome this difficulty we also had to generalize the kind of hierarchy that we can build on the set of quasiclosures. First, we needed to determine what kind of relations we were interested in uncover, and how did we want to quantify them. The following ideas guided our choice of algorithm:

- Two sets should be connected only if their intersection is not empty.
- The more of a set A is contained in a set B, the stronger the relation from A to B
- The bigger the set B is compared to A, the lesser the part of A that should be contained in B to have a strong relation going from A to B. In other words, a very big set will attract smaller ones even if their intersection is not that large.
- Two sets that have a mutual strong relation are considered equivalents, unless one is contained in the other, in which case the biggest one is a parent of the other in the quasihierarchy.

Having all of those ideas in mind, we developed the following algorithmic procedure, that takes as input a $degree$ and a $threshold$, and allows us to structure any kind of pretopological space:

- We select a list of sets of cardinality $degree$.
- We iteratively apply the pseudoclosure to each of those sets until they stop growing.
- We quantify the relation between each pair of sets with non empty intersection.
- We say there's a link in the quasihierarchy when the value of the relation is above the $threshold$.
- Sets that have links going in both direction are considered equivalent and one is selected randomly.
- The resulting quasiclosures with the respective links determine the quasihierarchy.

An image that shows the difference between this procedure and the structural analysis previously presented can be seen in 5.5, and the pseudocode of the algorithm is found in 5.4. There, we also need to specify the variable $type$ (either $short$ or $random$), in order to choose which method should be used to selected the initial list of sets. It's clear that when $degree = 1$ and the space is of type V we will recover the structure of the Largeron-Bonnevay algorithm (no matter which $type$ or $threshold$ we choose), which justifies the fact of calling the quasihierarchy a generalization of the previous algorithm.

Algorithm 5.4 Quasihierarchy

```
Require: A pretopological space $(U, a(.)), degree, type, threshold$
procedure QuasistructuralAnalysis($(U, a())$)
    $QF_e \leftarrow$ ElementaryQuasiclosures($(U, a(.)), degree, type$)
    $Adj_{qh} \leftarrow$ ExtractAdjacencyQuasihierarchy($QF_e$)
    $Quasihierarchy \leftarrow$ ExtractQuasihierarchy($QF_e, Adj_{qh}, threshold$)
end procedure

procedure ElementaryQuasiclosures($(U, a()), degree, type$)
    $SeedsList \leftarrow []$
    for all $x \in U$ do
        if $type = geometrical$ then
            $Seed \leftarrow$ ShortSet($x, degree$)
        else
            $Seed \leftarrow$ RandomConnectedSet($x, degree$)
        end if
        $SeedsList.append(Seed)$
    end for
    $QF_e \leftarrow$ ElementaryClosedSubsetsNV($SeedsList$)
    return $QF_e$
end procedure

procedure ExtractAdjacencyQuasihierarchy($(QF_e)$)
    $Adj_{qh} \leftarrow SquaredMatrixZeros(size(QF_e))$
    for all $F, G \in QF_e$ do
        $F\_has\_G \leftarrow Size(F \cap G)/Size(G)$              ▷ How much of G has F?
        $G\_has\_F \leftarrow Size(F \cap G)/Size(F)$              ▷ How much of F has G?
        $F\_bigger\_G \leftarrow Size(F)/Size(G)$           ▷ How much bigger is F than G?
        $G\_bigger\_F \leftarrow Size(G)/Size(F)$           ▷ How much bigger is G than F?
        $Adj_{qh}[Index(G), Index(F)] = G\_bigger\_F \cdot G\_has\_F$
        $Adj_{qh}[Index(F), Index(G)] = F\_bigger\_G \cdot F\_has\_G$
    end for
    return $Adj_{qh}$
end procedure

procedure ExtractQuasihierarchy($(QF_e, Adj_{qh}, threshold)$)
    $Adj_{qf}[Adj_{qf} < threshold] \leftarrow 0$
    $Adj_{qf}[Adj_{qf} < threshold] \leftarrow 1$
    for all $i, j \in Range(Size(U))$ do
        if $Adj_{qf}[i, j] = Adj_{qf}[i, j]$ then
            $Select\ one\ randomly\ and\ erase\ the\ other$
        end if
    end for
    return $Adj_{qf}$
end procedure
```

```
**procedure** ShortPath($FirstNode, degree$)
    $Path \leftarrow []$
    $LastNode \leftarrow x$
    **for all** $i \in range(degree - 1)$ **do**
        $NewNode \leftarrow ClosestNode(LastNode)$
        $Path.append(NewNode)$
        $LastNode \leftarrow NewNode$
    **end for**
    **return** $Path$
**end procedure**

**procedure** RandomPath($FirstNode, degree$)
    $Path \leftarrow []$
    $LastNode \leftarrow x$
    **for all** $i \in range(degree - 1)$ **do**
        $NewNode \leftarrow RandomNeighbor(LastNode)$
        $Path.append(NewNode)$
        $LastNode \leftarrow NewNode$
    **end for**
    **return** $Path$
**end procedure**

```

The pseudocode shows the calculation of the elementary quasiclosures as an extension of the originnal **ElementaryClosedSubsets**() procedure, *i.e.* by taking every subset and applying the pseudoclosure repeatedly until it stops expanding. Obviously we can also use the new version of the algorithm that we introduced above. An example of the difference in performance for some real datasets, with $degree = 4$ and $type = short$ and $threshold = 0.5$ was shown in 5.1.

We will see in the last chapter of the thesis how we can use these notions to treat problems of clusterization.

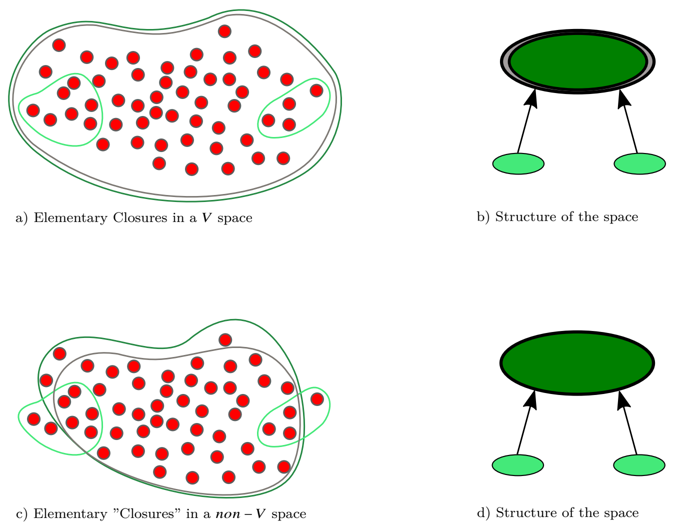

a) Elementary Closures in a V space
b) Structure of the space
c) Elementary "Closures" in a $non-V$ space
d) Structure of the space

Figure 5.5: Structures in a space

Schematic description of the construction of the Largeron-Bonnevay and the quasihierarchy structures.

Chapter 6

Learning a Pretopological Space

6.1 Definition

A very interesting use of pretopology has been presented recently, where the concept of learning a pretopological space (LPS) was introduced. There, the authors have completely switched the question, and instead of imposing a pretopology to study the information that pretopological concepts can provide, they have assumed that a pretopology is the natural characterization of a set, that it can be expressed as a combination of known networks, and that the elementary closed subsets are observations we can make: under all these assumptions, the question is now how to combine the known networks in order to get the correct pretopology (*i.e.* the one that has the known elementary closed subsets). The approach could be assimilated in spirit to the one used with so much success on deep learning.

The first work to propose this approach was [41]. There, the authors began by defining a parametrized pretopological space (**P-Space**), as a $V-type$ space $(U, a(.), \mathbf{w})$, where the pseudoclosure $a(.)$ is defined by

$$\forall A \in \wp(U), a(A) = \{x \in U \mid \sum_{N_k \in \mathcal{N}} \omega_k \cdot \mathbb{1}_{N_k(x) \cap A = \varnothing} \geqslant \omega_0\}$$

$such\, that$ (1) $\omega_0 > 0$, (2) $\sum_{k=1}^{K} \omega_k \geqslant \omega_0$ and (3) $\forall k, \omega_k \geqslant 0$.

K is here the number of different neighborhoods. Although the authors talk in terms of neighborhoods, as in the prefilter basis definition, they only consider prefilter basis with an equal number of members for every element of the set U, so in practice each neighborhood is seen as the neighbors of the element in one particular network.

With $\omega_0 = p$ and the rest of the $\omega_k = 1$, we can recognize a formalization of the idea advanced in [23] (*c.r. structural analysis* 5.1), where the parameter p indicates a requirement on the minimum number of neighborhoods that must intersect a subset A in order to expand it.

The authors used this framework to extract a lexical taxonomy. They began by taking a corpus of documents and applied four different techniques to extract the relation $R \mid xRy \iff x\, is\, more\, general\, than\, y$ out of those documents. They combined those networks with different weights in order to get a **P-Space** where the structural pseudoclosure proposed by Largeron-Bonnevay was as similar as possible to some manually established datasets extracted from WordNet. They used genetic algorithms with the $F - score$ as function of fitness for a population. Every time they created a new population, they calculated the elementary closures and they compare them to the expected ones in order to see how fit the population was.

A second work in the same subject was recently published [36], where some modifications were made to the previous framework. The first of those modifications was to change from a weighted set of networks to a DNF in terms of the networks similar to the one we presented here. The authors proved that a DNF is both more expressive and less redundant than the weighted version. On the one hand, the pseudoclosure obtained by any combination of weights can be emulated by a particular DNF, but not the other way around; on the other hand, many weights combinations are equivalent, increasing unnecessarily the space of possible answers that needs to be explored by the optimization algorithm.

Besides the change in framework, the paper proposes two ameliorations to the previous work. The first is to change the optimization method, from a genetic algorithm approach to a greedy algorithm approach; the second was to change the metric used to measure the quality of an intermediary solution, from the use of the $F-score$ to a score derived from the Multiple Instances Learning framework.

Multiple instance learning (MIL) is a form of weakly supervised learning where training instances are arranged in sets, called bags, and a label is provided for the entire bag. In other words, for a simple binary classification case, instead of having instances individually labeled correct or wrong, we receive them in groups, and we only know if there's a correct instance in the group or not.

For the LPS case, the positive bags are sets of all the possible pseudoclosures that are consistent with a particular closure. The idea behind this is that the previous work only considered the information delivered by the final step of the expansion (*i.e.* the closure), while this approach allows to privilege pseudoclosures that are more likely than others, even when having the same final closure.

The complexity of the new algorithm is presented in terms of the number of times the whole structural process has to be calculated, and its equal to $O(max\_iter \cdot \mid V \mid \cdot beam\_size)$. Now, as we saw in the section dedicated to the Largeron-Bonnovay algorithm, even one structural analysis might be extremely slow in some cases, let alone a multitude of them. We will present in the third section of this chapter a new algorithm that gives an exact answer to the problem (when it exists) in a fraction of the time of a single structural analysis calculation.

6.2 Difference with our framework

Before presenting our algorithm to solve the LPS problem, we would like to talk about the differences between the **P-Space** as presented in the cited papers and our framework, since they might look similar at first sight.

Although both frameworks use the notion of a $DNF(.)$ (our choice was directly motivated by their work), the concept of **P-Space** is still too closely related to the notion of neighborhoods, and in consequence is still very rigid.

Without going into the details again (*c.r. section Equivalences: subsection Multilayer valued relationships*), we know that our framework is capable of generating non $V-type$ pretopological spaces, something that might become very useful when mixed with the quasihierarchy concept.

Another simple way to see how our framework is more flexible, is by noticing that although we still use a positive $DNF(.)$, it's easy to simulate negation by inverting the signs of a network G_i and its threshold θ_i. Indeed, to ask for an element x that it doesn't belong to the pseudoclosure of A in network G_i is to say that the sum of the edges going from x to A is less than θ_i. This is the same as asking that the sum of the additive inverses of the weights are bigger than $-\theta_i$, which is to ask x to belong to the pseudoclosure of $-G_i$ with $-\theta_i$ as threshold.

It's important to realize our framework is not just a pragmatic numerical device, but has a natural interpretation. When the idea of the parametrized pretopological space was introduced, the weights assigned to each network were interpreted as a quantification of the credibility that one was giving to the network. When considering not only if the neighbors of an element intersect a set, but also the strength of the intersection, we are giving a network the possibility to be more credible in some zones than others. A network can be more credible than other, but if the other manifests a really strong desire to join an element to the pseudoclosure, attention should be paid. Credibility becomes then a local issue, but the interpretation remains valid.

As the reader might have appreciated, our framework is nothing but a mixture of the two most general frameworks presented so far: the multilayer valued relations introduced in [91], and the parametrized pretopological space introduced in [36]. Putting the two together though, allows us to create pretopologies that none of them could generate on their own.

6.3 Biggest Smaller Pseudoclosure Algorithm

We will finish this part of the thesis presenting an algorithm that will allow us to estimate a $DNF(.)$ with the exact same closures we are trying to reproduce, when that $DNF(.)$ exists.

The idea is the following: let's suppose an element x is external to the closure C, but belongs to $a(C)$ on graphs G_1 and G_2 (*i.e.* the sum of its connections to C in G_1 is bigger than θ_1, and the sum of its connections to C in G_2 is bigger than θ_2)). Then, having both $V_1(.)$ and $V_2(.)$ equal to **True** is not sufficient to propagate, so the conjunction $V_1(.) \wedge V_2(.)$ is not part of the $DNF(.)$ that defines the pretopology.

If we were to apply that procedure with every closure C_i we are trying to reproduce, and every element external to C_i, we will end up with the set of all conjunctions that we are sure are not part of the $DNF(.)$, we will call this set the *forbidden conjunctions.*

The next thing to realize is that if a conjunction $\wedge_{iinI} V_i(.)$ is part of the *forbidden conjunctions*, then any conjunction that includes a subset of those boolean functions can't be a part of the $DNF(.)$ either. For example, if $V_1(.) \wedge V_2(.) \wedge V_3(.)$ doesn't propagate when equal to **True**, then certainly $V_1(.) \wedge V_2(.)$ is not enough to propagate either. The Hasse diagram of the sets of boolean functions used in each conjunction can be seen in 6.3.

Once we have figured out which conjunctions are not part of the $DNF(.)$, we could take the disjunction of all the other conjunctions as the estimated $DNF(.)$, and we would definitely get the closures we were trying to retrive if a solution really exists. The reason is that we would be expanding as far as possible without ever crossing the boundaries established by the closures. But a lot of those conjunctions would be unnecessary. Indeed, the case is the opposite than before, if we already have a set of boolean variables in a conjunction, another conjunction that includes a superset of those variables would be redundant, since it will be **True** only if the first one is **True**. In conclusion, theres's no need to include a superset of a conjunction that we already included.

The pseudocode of the algorithm that uses those two ideas, in order to retrieve the shortest (in terms of number of conjunctions) $DNF(.)$ that will expand every set as far as possible, without transpassing the closures, is given in 6.3.

```
procedure BiggestSmallerClousure($(U, a(.)), closures\_set$)
    $forbidden\_conjunctions \leftarrow$ ExtractForbiddenConjunctions($(U, a(.)), closures_set$)
    $DNF \leftarrow$ ExtractDNF($forbidden\_conjonctions$)
end procedure

procedure ExtractForbiddenConjunctions($(U, a()), closures\_set$)
    for all $C \in closure_set$ do
        $forbidden\_conjunctions \leftarrow list()$
        for all $x \in closure\_neighbors$ do        ▷ external elements connected to the closure
            $conjunction \leftarrow list()$
            for all $G_i \in (U, a(.))$ do
                if $V_i(A, x) =$ True then
                    $conjunction.append(V_i)$
                end if
            end for
            $forbidden\_conjunctions.append(conjunction)$
        end for
    end for
    return $forbidden\_conjunctions$
end procedure

procedure ExtractDNF($forbidden\_conjunctions$)
    $dnf \leftarrow list()$
    for all $conjunction$ do        ▷ Every possible conjunction, seen as a set of $V_i$'s
        $conjunction \leftarrow list()$
        for all $forbidden \in forbidden\_conjunctions)$ do
            if $conjunction \subset forbidden$ then
                $next\ conjonction$
            end if
        end for
        for all $included \in dnf)$ do
            if $included \subset conjunction$ then
                $next\ conjunction$
            end if
        end for
        $dnf.append(conjunction)$
    end for
    return $dnf$
end procedure
```

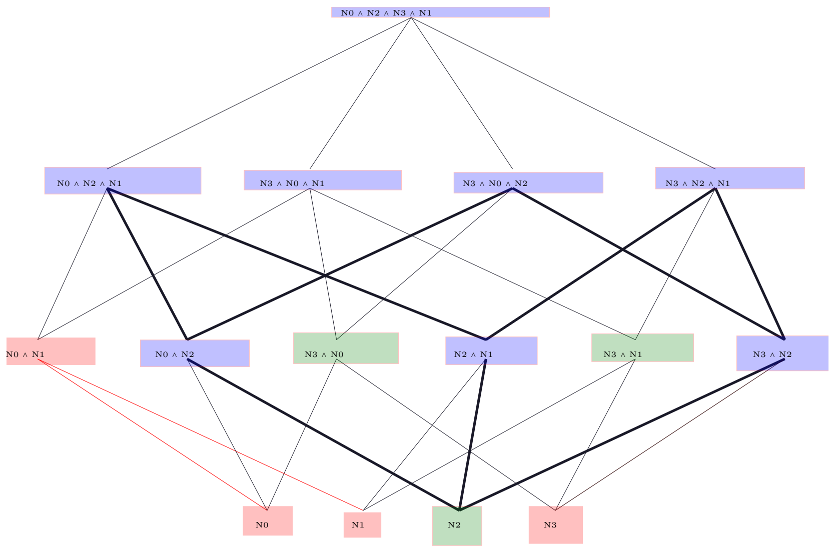


Figure 6.1: Biggest smaller pseudoclosure algorithm.

Example for the case where neither $N_1 \wedge N_2$ nor N_3 propagate. When in red, it disqualifies the children, also in red; when in green, it disqualifies the parents, in blue. We only keep the green ones.

Part III

Applications

Chapter 7

Python Library

As it was mentioned in the introduction, although many works use the concepts of pretopology, their implementation is rarely discussed, and it's usually made from scratch and in an ad-hoc manner. Lamure [82], for example, describes his implementation in great detail, but it's very specific to the case of image processing.

A first attempt to undertake this problem was done in 2010 with the creation of a JAVA library called pretopolib [92]. Although the library had many interesting features, it seems to be discontinued, and it didn't reach much popularity.

Besides those practical considerations, two main reasons pushed us to start the task from the beginning. The first one is pragmatical; although JAVA is faster, and still quite popular in production, Python has without a doubt become the de-facto language for the study of complex systems and machine learning. It is extremely easy to use, so many researchers that are not computer scientists and are only interested in studying a particular problem, begin by learning python. The size and dynamism of the community is enormous, and the number of available packages is extremely large and keeps growing.

The second motivation is more theoretical. Pretopolib was mostly built around the notion of a set, with the JAVA implementation of it. The implementation was based on hashtables to store the information, which allowed to answer questions about the membership of an element to a pseudoclosure in amortized constant time. A lot of care was put into the proper choice of data structures to deal with different kinds of pretopologies, but they all were related to sets. It is true that under that conception of the space, every pretopology could be modeled, but the way of dealing with the issue was so general that there was little place for optimization.

Most of the efforts were put into optimizing data structures so they could retrieve the pseudoclures of sets that had already been calculated. This seems like a strange choice to us, since even if we were able to store all the pseudoclosures calculated (which is impossible even for relatively small sets); the amount of time necessary to calculate them in the first place would be too large.

In order to avoid this problem, the author of the library also proposed a ”dynamic pseudoclosure”, one that was calculated from a set of multiple relations only at the time of being used. This was similar in spirit to what we are proposing here, but although the space could store many relations, the pseudoclosures seemed to be defined only for one relation at a time and there was no discussion about how to mixed them up.

All these considerations motivated us to create a new pretopological library made in Python and conceived from the beginning to deal with pretopologies described as our framework proposes. The library was named pretologyx, because of the heavy use that it makes of the networkx library.

In the next section we will present the different modules of the library in a general manner, in order to get an idea of how they are supposed to interact with one another, and what are the tasks that can be accomplished. We will also describe some implementation choices.

In the appendices we will find more details about the different functions and the arguments that they take.

7.1 Modules

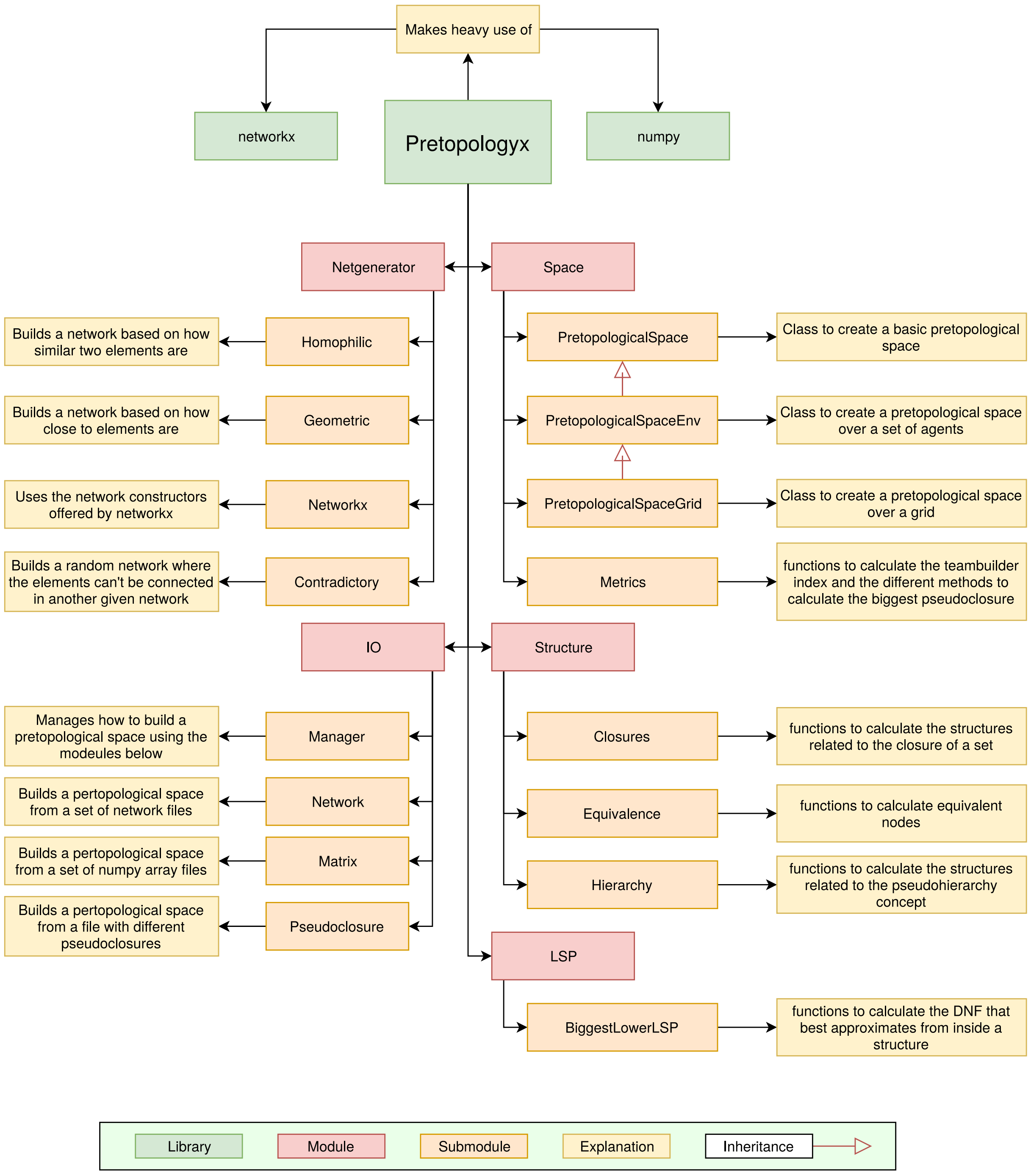

Makes heavy use of
networkx
Pretopologyx
numpy
Netgenerator
Space
Builds a network based on how similar two elements are
Homophilic
PretopologicalSpace
Class to create a basic pretopological space
Builds a network based on how close to elements are
Geometric
PretopologicalSpaceEnv
Class to create a pretopological space over a set of agents
Uses the network constructors offered by networkx
Networkx
PretopologicalSpaceGrid
Class to create a pretopological space over a grid
Builds a random network where the elements can't be connected in another given network
Contradictory
Metrics
functions to calculate the teambuilder index and the different methods to calculate the biggest pseudoclosure
IO
Structure
Manages how to build a pretopological space using the modeules below
Manager
Closures
functions to calculate the structures related to the closure of a set
Builds a pertopological space from a set of network files
Network
Equivalence
functions to calculate equivalent nodes
Builds a pertopological space from a set of numpy array files
Matrix
Hierarchy
functions to calculate the structures related to the pseudohierarchy concept
Builds a pertopological space from a file with different pseudoclosures
Pseudoclosure
LSP
BiggestLowerLSP
functions to calculate the DNF that best approximates from inside a structure
Library
Module
Submodule
Explanation
Inheritance

Figure 7.1: Library Structure

We will start by explaining some basic features that are common to all modules. In particular, most of the heavy calculations use NumPy, which is the fundamental package for scientific computing with Python.

A set in our library will always be represented by a NumPy array the size of the space, with ones representing membership, and zeros otherwise. This specification is certainly sub-optimal from a memory space point of view. Still, the increase in performance, compared to any Python implementation involving loops, is so big that the exchange is well worth it. Indeed, due to its flexibility Python can be an extremely slow language. The language is not typed, and lists can be created with any kind of different objects inside. A loop in Python will therefore create a series of dummy variables, and will have to test a number of things on the elements of the list before performing any operation on them. NumPy arrays, on the other hand, will contain values of the same type, and they all will be placed consecutively in the memory. Operations on that array will be executed directly in C instead of the Python level for loops.

The same idea motivated our choice for network representation: each network will be described by its weighted adjacency matrix. Besides the advantage in performance, this allows to work with sets and networks using the same type of representation. The importance of this cannot be overstated: the fact that our framework uses networks to describe a pretopology, which is a set theoretical notion, makes it is necessary to switch back and forth constantly between those two notions. Being able to use the same mathematical object to describe them both, greatly facilitates the task.

The pseudoclosure operator, for example, which is the most important notion of the theory, becomes a simple multiplication of the adjacency matrix by the membership vector, and a comparison of the resulting vector with the threshold.

Multiple network pretopologies involve the same process multiple times, one for the pseudoclosure of each network. Then, we need to make some element-wise multiplications or sums of the resulting vectors, in order to get their intersection or union and evaluate the $DNF(.)$. All of these operations are done extremely fast by NumPy.

Now that we know how do we represent the basic notions in the library, we can describe the different modules. An image of the whole structure of the library is shown in figure 7.1.

7.1.1 Space

7.1.1.1 Prenetworks

The library uses the concept of a Prenetwork, which is characterized by one network (in the form of its weighted adjacency matrix) and a list of tuples $(threshold, weight)$ associated to it. A Prenetwork with adjacency matrix A and list of tuples [(1, 1),(2, 1),(1,2)], is equivalent to three networks with adjacency matrices A, A and 2A, and thresholds $\theta_1 = 1$, $\theta_2 = 2$ and $\theta_3 = 1$. This not only allows us to store multiple networks saving a lot of space, but also the pseudoclosure can be calculated much faster. Indeed, the matrix multiplication will be done only once, and the resulting vector is the one that will be multiplied by the different weights and compared to the different thresholds.

The need to specify pretopologies defined over multiple networks in that manner, was motivated by the LPS problem. We already showed how to learn a pretopological space over a specific set of networks previously defined. For some problems though, we might not know which are the networks defining the space, or what thresholds should be associated to them. The concept of Prenetwork should allow us to test multiple different options, and let the learning process find out which are the right parameters.

7.1.1.2 Pretopological Spaces

A pretopological space is represented by the **PretopologicalSpace** class. The way of storing the space is that proposed by our framework in section three 3.1, so we need to save a set of networks with thresholds associated to them, and a $DNF(.)$. The networks and their thresholds are stored in the form of **Prenetwork** objects, while the $DNF(.)$ will be saved as a list of lists, with each inner list containing the indices of the networks that form a single conjunction.

Since we are interested in establishing pretopology as a way to study complex systems, and we believe -as we will argue in the next section-, that agent-based models are a great laboratory to reproduce those systems, we have tried to facilitate the communication between pretopology and agent-based models. We have concentrated our efforts in designing an easy to build relation between agent populations and pretopological spaces. For this, the populations should be described as a 2D array where each row is an agent and each column is an attribute.

This population is seen as an environment over which a pretopological space will be built. Two specifications of the class **PretopologicalSpace** were designed, each inheriting (in an Object Oriented sense) from the previous one.

- The most general of those two is the class **PretopologicalSpaceEnv**, which includes a *population* attribute besides those that a general **PretopologicalSpace** has. The idea is to facilitate the creation of networks where links between agents depend on their attributes, and then use those networks to define a pretopological space.

  The types of networks that can be built using population attributes are described in the Network module.

  This type of space was used in the application *target selection* 8.2

- The other specification is the **PretopologicalSpaceGrid**, which inherits from **PretopologicalSpaceEnv**. The particularity of this space is that the population can be arranged in the form of a grid. The agents of the population are perceived as the cells of the grid, and the kind of neighborhoods that we can define are those presented in 3.2.1.2

  This type of space includes a method to draw the grid, with the color of each cell dependent on some attribute of the population.

  This type of space was used in the application *learning propagation* 8.3

7.1.2 Structure

The library implements the different structural algorithms that were discussed during the thesis. The methods receive a **PretopologicalSpace** and calculate:

- The elementary closed sets using the Largeron-Bonnovay method 5.1.
- The elementary closed sets using our new method 5.1.
- The degree n generalization of those methods. 5.1.
- The structural analysis using any of the previous methods. 5.1.
- The strongly connected version of equivalence structure. 5.2.
- The quasiclosures of a space. 5.3.
- The quasihierarchy of a space. 5.3.

7.1.3 LPS

The library implements the *biggest smaller pseudoclosure* method to learn a pretopological space that was described in section 6 6. The method receives a **PretopologicalSpace** and a list of sets (the closures) as input, and estimates the $DNF(.)$ that will make the closures of the **PretopologicalSpace** best approximate the closures given as input.

7.1.4 IO

Two modules are in charge of building or storing pretopologies. One from files, one during execution. The first one is the IO module, the second is the Network module.

7.1.4.1 Graph inputs

The IO module allows to easily transfer a pretopology between computers by saving it in a particular format. More importantly though, is that it helps us build pretopological spaces from files with real networks. This is another advantage of working with networks. The library *networkx* has a great many methods to read networks from external files. If we put those network files in a single folder, and we include two other files *threshold_weights* and *dnf*, the library is able to build a pretopological space from that folder. The *threshold_weights* file describes the different lists of $(threshold, weight)$ tuples associated to each network file. The *dnf* file is a list of lists, each of the inner lists enumerates the indices of the networks belonging to a same conjunction.

7.1.4.2 Other inputs

Besides the files describing a graph that are supported by networkx, the networks that constitute the pretopology can be described by:

- The adjacency matrix of the graph stored as a an *.npy* file. This is the format used by NumPy to store arrays.
- A file with a list of tuples $(set, pseudoclosure)$. Both sets and pseudoclosures will be described by the list of indices of the elements. The method to build the pretopology is described in 3.2.3
- A file with a list of lists describing the prefilter basis. Each inner list will have the indices of the elements belonging to a neighborhood. The method to build the pretopology is described in 3.2.1.1

7.1.5 Network

This modulo allows to build networks that are used to define a pretopological space. The same as the IO module, networks can be built either using Networkx, or the following three additional methods that work for **PrestopologicalSpaceEnv** objects:

- Geometrical: we specify a *threshold* and an *attribute*, and every pair of agents whose euclidean distance for that attribute is smaller than the threshold are connected.
- Homophilic: we specify an *homophily_level* and an *attribute*, and we connect agents according to the algorithm described in 8.1.

- Contradictory: we specify an existent network and a network creation method, and a new network is created using that method but preventing any links between nodes connected in the input network. This allows the creation of networks of friends and enemies, for example.

Chapter 8

Diffusion

This chapter uses pretopology to address two problems related to diffusion phenomena. Each problem uses the theory in a different form: the first as a way to structure a space and uncover central groups; the second as a way of modeling a dynamic process.

The first problem is that of selecting a group of elements that will maximize the spread. To put things in a more concrete form: if we are studying the diffusion of a new opinion inside a population, the problem consists in finding the group of people that will spread the most this new opinion over time. We use some pretopological metrics to select the initial set of people, and we compare their performance with some other classic selection strategies.

The second problem uses pretopology to model the diffusion of a fire in a forest. Each application of the pseudoclosure function gives the set of burnt zones in a new time step. The problem is that of learning the pseudoclosure that models the fire diffusion, given some areas that do not spread the fire. We use the exact same settings used by the paper that introduced the problem, and we test the results we presented in chapter six to see if they perform according to our predictions.

Both problems use agent-base models as a simulation tool, so we begging by describing what they are. We then give the details and results of the target selection problem, and we finish with the details of the LPS problem.

8.1 Agent-based Models

Agent-based models are computational models mostly used for the study of complex systems by the use of multiple simulations [67, 54, 95]. The models consist of a set of virtual entities called agents that evolve and interact inside an environment.

An agent is a virtual object that usually represents an entity from the real world (*e.g.* a person, an animal, a car, etc...). They might be defined by the following three sets:

- A set of attributes: characteristics of the agent that do not change during the simulation (*e.g.* the gender of a person).
- A set of variables: characteristics of the agent that evolve over time (*e.g.* the position of a person in its environment).
- A set of functions: descriptions of the actions that the agent must execute when certain triggering conditions are fulfilled. These actions might trigger some other actions or could change some variables' values.

Object Oriented Programming is therefor a very suitable paradigm for the design of agent based-models.

Sometimes agents are also described as having relations, but we consider these as a particular attribute or variable. It is an attribute when the connections are fixed over time, and a variable when the set of related agents emerges from the way the agent evolves in the environment.

We should not confuse agent-based models with multi-agent systems. The former are virtual systems where we program the behavior of the individual components to study what emerges globally, and the later are real physical systems of independent asynchronous components designed to carry out a task [51].

Although the line is somewhat fuzzy sometimes, we could say that agent-based models are divided into two categories according to their degree of realism and their motivation.

8.1.1 Realistic

Some of the models are extremely realistic: they have several different kinds of entities interacting in very sophisticated ways, attributes and variables are instantiated with real data, and their description may take hundreds of pages. These models are usually employed to help in public policy decision making in areas such as epidemics or transportation, or as a way to treat problems in operational research that would be hard to model otherwise.[14, 112, 72, 128, 18, 13, 43]

The origin of this kind of models is sometimes traced back to [106], under the name of microsimulation. Orcutt presented a model of a socio-economic system where individual units received inputs, treated them according to some probabilistic rules and generated outputs that were used by other units in the next time step.

From an historical point of view it might be interesting to realize that Von Neumann, who was fundamental in the development of game theory, was also fundamental in the development of computers. According to his daughter, Von Neumann thought game theory would be a much more popular tool for the study of social systems, and he was well aware of the limits of mathematical analysis to treat the problems of the theory. He is also known to have had the use of computers in game theory problems as an eventual goal, so it doesn't seem too far fetched to say he had some sort of agent-based model in mind.[75]

One important characteristic of agent based models is the natural way they have of mapping reality. Indeed, they describe the world very much as we conceive it in our daily lives [10, 9] the models are usually made up of agents that represent individuals who realize actions when meeting certain conditions; this is quite similar to the way natural language deals with the people and objects that surround us.

So even if agent-based models are perfectly formal, and their subtleties might escape some people, most of their content can be easily grasp by a person with no formal training. Compare this, for example, to the very technical and abstract vocabulary that is necessary for the comprehension of some dynamical system models of social behavior, not to mention physical models. This characteristic is quite important when the models are supposed to be used by a series of people coming from very different backgrounds.

8.1.2 Toy models

A second type of model, are those usually called *toy models*. These are much simpler in their premises, closer to particle physics models where we cannot get analytical results. In most of these cases the interest is not so much in the particularities of a singular initial configuration, but in the relation between the different individual behaviors and the resulting global patterns. Since these models are usually quite simple, they allow to make multiple simulations, with many sets of parameters, without using to many computational resources [97, 70, 124, 132, 114, 73].

The whole process is similar in spirit to the dynamic networks that we saw in the context section. The definition of individual behavior allows us to study the mechanisms that give origin to macro phenomena. It is only the type and the complexity of behavior that changes.

Toy models are interesting because the computer stops being a tool to solve problems whose solutions we could get otherwise if we had the time and manpower, and becomes a real laboratory for experimentation. Perhaps the most emblematic example of this is still Stephen Wolfgram, who shut himself away from the physical community to spend years exploring the different evolutions of one dimensional cellular automata. The results came out in the form of a thousand pages long book [134], and although it might not be as groundbreaking as he presents it, it is certainly unique in its genre. Even though the book is presented as a book on cellular automata, it is quite different from others on the subject, where formal languages, fractals and other mathematical tools are used in order to study analytically the evolution of the automata [74]; here, it is computer experimentation that is at the center of the book.

It is usual in these kind of agent-based models to talk about *emergent phenomena*, so we should say a few words about this concept.

8.1.3 Emergence

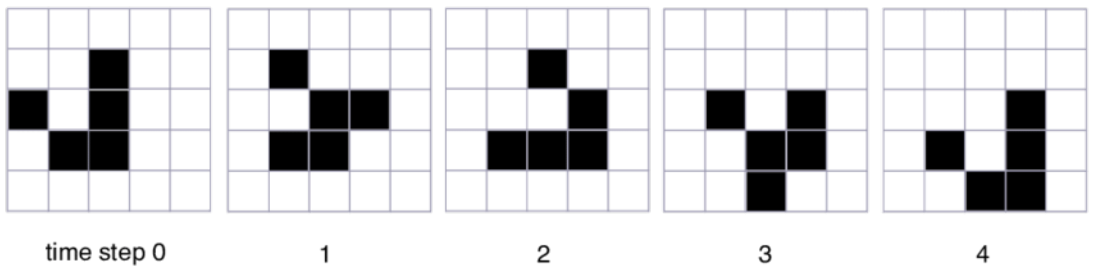
Although a few formal ways of defining *Emergence* have been proposed (information theory, formal languages), none of them has been able to impose itself as the proper one. The concept is usually defined in a loosely fashion as ”The sum is more than it’s parts” [102]. We believe this definition leads to confusion: it seems to convey some mystical connotations, and distorts the real potential of a model. If we put a million wooden pieces in the form of a circle, the whole would have a radius, while none of the units would, but there is nothing complex about the phenomenon and few people would call this emergence.

Here we have in mind what Chalmers calls *weak emergence* [40]. *Strong emergence* is when something fundamentally different in nature happens at the macro level. *Weak emergence*, on the other hand, is when the macro level exhibits a behavior that is at once comprehensible and very hard to predict.

The Conway’s game of life is still one of the best ways to illustrate what we mean. Here we begin with a grid with black and white squares, and the color of each square changes in discrete time steps according to some simple rules about their Moore’s neighborhood 3.2.1.2. Although the model is quite simple, extremely complex patterns emerge at the macro level: some of them seem to glide through the grid over time, and if disposed in a proper fashion are capable of reproducing the calculations of any Turing machine. It is obvious in this case that there is nothing at the macro level that was not already on the components, if we consider the rules of interaction as something proper to the component.

This difference may seem like a subtlety of little practical importance, but it has as consequence that the invention of new concepts can lead to new phenomena rendered intelligible, which in turn signifies a better understanding of the emergence from micro to macro. Now, the structural concepts we use are constrained by the structural framework chosen to represent the system. When we realize this, the role of the structural framework takes a whole new level of importance.

Going back to the network parallel, the emergence of a *power law* degree distribution or the small-world property are only evident once we have decided to use networks as a way to characterize the system.



The glider effect is a classical example of an emergent phenomena.

This is why when conceiving pretopologyx we were interested in facilitating the relation between agent-based models and pretopology.

8.2 Targets Selection

Problems related to diffusion are everyday more present in our lives (epidemics, misinformation spreading, computer viruses, etc...). To confront these problems the strategy has been to identify the individuals that are more central in the system in order for them to stop or accelerate the diffusion process. But that begs the question of what is to be central in a complex system. This question is strictly conditioned to the way the system structure is characterized, and the traditional way to deal with this has been by the use of networks [70, 132, 38, 73].

Here, we present the results that we published in [81]. We see that the notion of *teambuilder* helps us develop a greedy algorithm that has very good results finding the centralthe biggest pseudoclosure or the index are appropriate ways of identifying superspreaders. The relevance of these metrics will be tested by the use of some standard agent-based models of epidemics and opinion dynamics. Finally, a pretopological model of opinion diffusion will also be proposed and studied.

8.2.1 Diffusion Simulations

We will now introduce the different models that will be used to test how well the pretopology captures the idea of an agent being central in a process of diffusion. Although these models are presented as opinion dynamic models, they are so general that have been used as models of epidemics, segregation and other diffusion phenomena [66, 117, 19, 129]. The models are the following:

- A Cascade model: where at each time step, every agent that changed opinions in the previous step will try to convince each of its neighbors to also change opinion. The process of convincing is modeled by a Bernoulli trial with probability equal to the influence that the agent has over its neighbor.
- A Threshold model: where at each time step an agent will change opinion if the fraction of its neighbors that has the opposite opinion is superior to a certain threshold.
- A Utility-based model [73] where an agent will change opinion if the utility of changing is superior to zero. This will depend on two parameters, its *motivation* to change, and its *conformity* to accept the opinion of the others.

The pseudocode for each of these models, as well as for the whole methodology can be found in 8.1. The methodology consists in building a population, create a network on that population, select a target group that will have a different opinion than the rest, and execute an opinion dynamic model.

The creation of the network is done according to the same procedure used in [73]. This procedure allows to control for the degree of clustering of the network through a parameter called *homophily_level*, which at the same time captures the idea that people are usually connected to people that are similar to them[100].

The algorithm adds one edge at a time by selecting a random agent, and randomly deciding (with *homophily_level* probability) if the agent is connected to its most similar agent, or to an agent selected randomly. The pseudocode of this procedure can also be seen in 8.1.

For each value of the parameter *homophily_level* different networks are created (parameter *number_networks*); for each network, all selection strategies are performed; if the selection strategy for the targets is not deterministic, we select a target group and execute the opinion dynamic models multiple times (parameter *number_selections*). The results are stored and averaged over all networks that used the same *homophily_level*, and over all the selections for the non-deterministic strategies.

It's important to notice that only in the Utility-based model the *homophily_level* is really modeling the concept of homophily. Indeed, the algorithm for network creation is using the *motivation* of each agent as their similarity attribute (*c.r.* bellow), but none of the other models use that variable. So for those other two models the *homophily_level* will only control the structural clustering of the network.

Each simulation starts with the targets having opinion 1 and the rest of the population having opinion 0.

The parameters for the whole process are the following:

- population_size = 1000
- average_neighbors = 10
- number_networks = 15
- number_selections = 15
- homophily_levels = [0, 0.2, 0.4, 0.6, 0.8, 1.0]
- targets_size = 50

And the selection strategies are:

- Largest motivation
- Smallest motivation
- Largest *teambuilder* index
- Smallest *teambuilder* index
- *gti* heuristic: we test an heuristic for selecting a large pseudoclosure by selecting a random node x_1, and then selecting the node x_2 that maximizes $gti(x, \{x_1\})$. Subsequently we add the node x_3 that maximizes $gti(x, \{x_1, x_2\})$. The process continues until we have selected the number of targets necessary.
- Largest eigenvector centrality
- Largest pagerank centrality
- Random pseudoclosure: we take many random sets of targets, we measure the size of the their pseudoclosures, and we select the one with the largest one. The number of random sets selected was fixed to 500 in our simulations; the motivation being to spend around three times the amount of time spent with the *gti*() heuristic.
- Largest betweenness centrality
- Largest closeness centrality

8.2.2 Analysis

8.2.2.1 Individual network models

The first thing we notice is that the more clustering we have, the more different the metrics are among them, with most of them performing in a very similar manner when the network is completely random, but *degreecentrality*, *pagerank* and *gti* standing out as the homophily level increases, and *gti* working systematically better for the cascade and threshold model.

Algorithm 8.1 Diffusion Simulation

```
procedure METHODOLOGY($(parameters)$)
    $agents \leftarrow$ CREATEAGENTS($population\_size$)
    for all $hl \in homophily\_levels$ do
        for $i \leftarrow 1$ to $number\_populations$ do
            $network \leftarrow$ HOMOPHILICNETWORK($agents, attribute, hl$)
            $prenetwork \leftarrow Prenetwork(network, threshold = 1)$
            $pre\_space \leftarrow PretopologicalSpaceEnv(prenetwork, dnf = [0])$
            for all $target\_strategy \in target_strategies$ do
                $targets \leftarrow$ SELECT_TARGETS($target\_strategy$)
                $results\_cascade \leftarrow$ CASCADE_DIFFUSION($pre\_space$)
                $results\_threshold \leftarrow$ THRESHOLD_DIFFUSION($pre\_space$)
                $results\_utility \leftarrow$ CASCADE_DIFFUSION($pre\_space$)
            end for
        end for
    end for
end procedure

procedure CREATEAGENTS($population_size$)
    for $i \leftarrow 1$ to $population\_size$ do
        $agent_i.motivation \leftarrow random(-1, 1)$          ▷ used by the utility model
        $agent_i.conformity \leftarrow random(0, 1)$           ▷ used by the utility model
        $agent_i.threshold \leftarrow random(0, 1)$            ▷ used by the threshold model
    end for
end procedure

procedure HOMOPHILICNETWORK($(attribute, hl)$)
    for $i \leftarrow 1$ to $average\_neighbors * population\_size$ do
        $n_1 \leftarrow randomNode()$
        $rand \leftarrow random(0, 1)$
        if $rand < hl$ then
            $n_2 \leftarrow closestNode(n1, attribute)$      ▷ closest agent w/r attribute value
        else
            $n_2 \leftarrow randomNode()$
        end if
        $edge \leftarrow addEdge(n_1, n_2)$
        $edge.influence \leftarrow random(0, 0.02)$          ▷ Each edge has an influence
    end for
end procedure

```

When interpreting this result in the context of a social system and a marketing strategy, for example, it becomes interesting to realize that the degree of an agent in the world (e.g. how much ”followers” he/she has) usually correlates to a bigger price or effort to convince him/her to adopt a new strategy.

In contrast, the pretopological approach does not necessarily take agents that are individually the most powerful, but those that as a group are influential, so the individuals in the group may be cheaper or easier to persuade.

The *teambuilder* index, on the other hand, while not irrelevant to the amount of spreading (as can be seen by the difference between the targets with the largest and smallest *teambuilder* indices) , did not perform particularly well as a strategy for select-

```
procedure CASCADEDIFFUSION(())
    for all $t \in steps$ do
        for all $agent \in agents$ do
            if $agent.opinion_{t-1} == 0 \wedge agent.opinion_t == 1$ then        ▷ just changed opinions
                for all $neigh \in agent.neighbors$ do
                    if $neigh.opinion_t == 0$ then
                        $rand \leftarrow random(0, 1)$
                        if $rand < edge[agent, neigh].influence$ then
                            $neigh.opinion_{t+1}$
                        end if
                    end if
                end for
            end if
        end for
    end for
end procedure

procedure THRESHOLDDIFFUSION(())
    for all $t \in steps$ do
        for all $agent \in agents$ do
            if $sum(agent.neighbors.opinion == 1)/|agent.neighbors| > agent.threshold$ then        ▷ neighbors fraction with opinion 1
                $agent.opinion = 1$
            else
                $agent.opinion = 0$
            end if
        end for
    end for
end procedure

procedure UTILITYDIFFUSION(())
    for all $t \in steps$ do
        for all $agent \in agents$ do
            if $agent.conformity * (1 - 2 * sum(agent.neighbors.opinion == 1)/|agent.neighbors|) + (1 - agent.conformity) * agent.motivation > 0$ then        ▷ The utility of 1 is bigger than that of 0
                $agent.opinion = 1$
            else
                $agent.opinion = 0$
            end if
        end for
    end for
end procedure

```

ing the targets.

Another thing that we can notice is how correlated are the size of the pseudoclosure with the final propagation of the opinion, proving that under the right circumstances it should suffice to study the structure of the population to get a good idea of their dynamics.

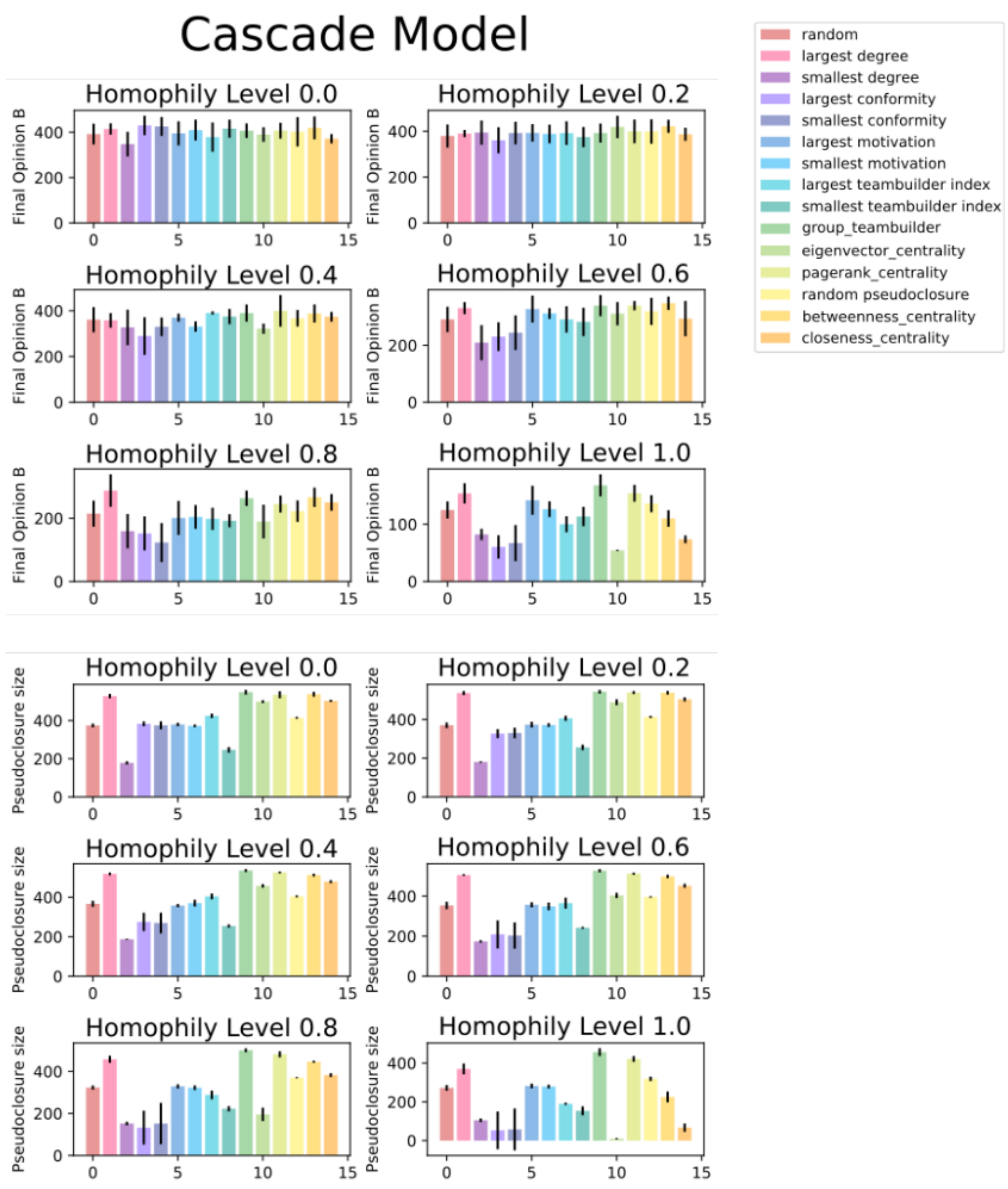


Figure 8.1: Results of the cascade model of diffusion

A final interesting result is that the *gti* heuristic works consistently better at finding big pseudoclosures than $random_pseudoclosure$, although the number of random samples had been chosen so it takes around three times more to select the targets randomly than with *gti*.

The interest of finding good heuristics for the problem of finding large pseudoclosures becomes obvious when we realize that no polynomial time algorithm may exist unless P=NP. Indeed, we can see that by thinking that if we had a polynomial time algorithm for finding the biggest pseudoclosure, we could apply it to the sets of a single element, then to those of two elements, and follow until the biggest pseudoclosure would include the whole space; this would give a polynomial time algorithm for the minimum dominating set problem, a problem well known to be NP hard.

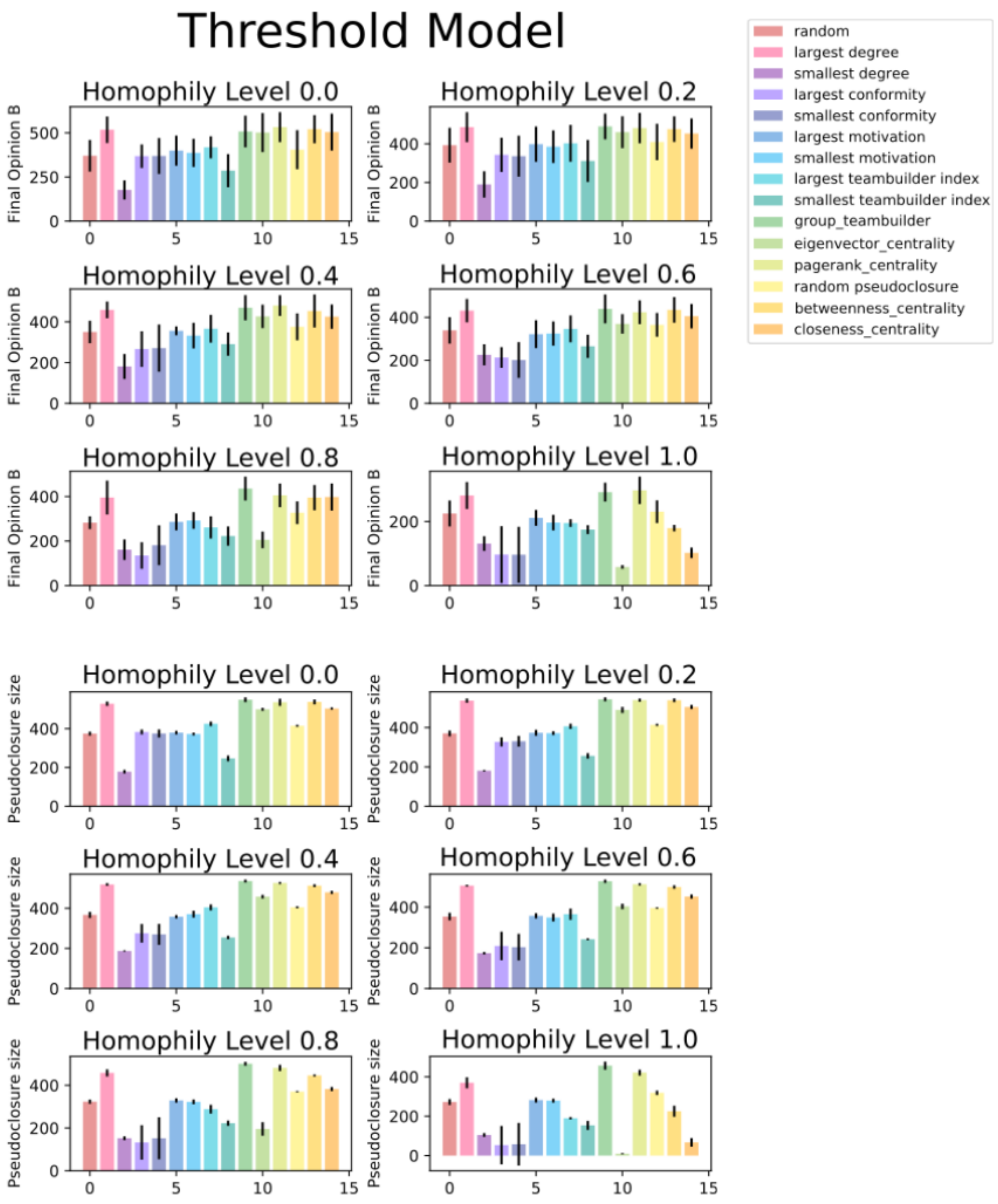


Figure 8.2: Results of the threshold model of diffusion

8.2.2.2 Mixed model

A second type of model was designed so a more sophisticated pseudoclosure could be used. The model is exactly the threshold model, but instead of just one network, we added a second one, where nodes would select three elements at random (as long as they are not connected in the first network), and they consider them their enemies. Under this model a person changes opinion if the sum of his/her friends with the new opinion, minus the enemies with the new opinion, divided by the total of connections, is bigger than the threshold.

We defined a second pseudoclosure function, where someone will belong to the

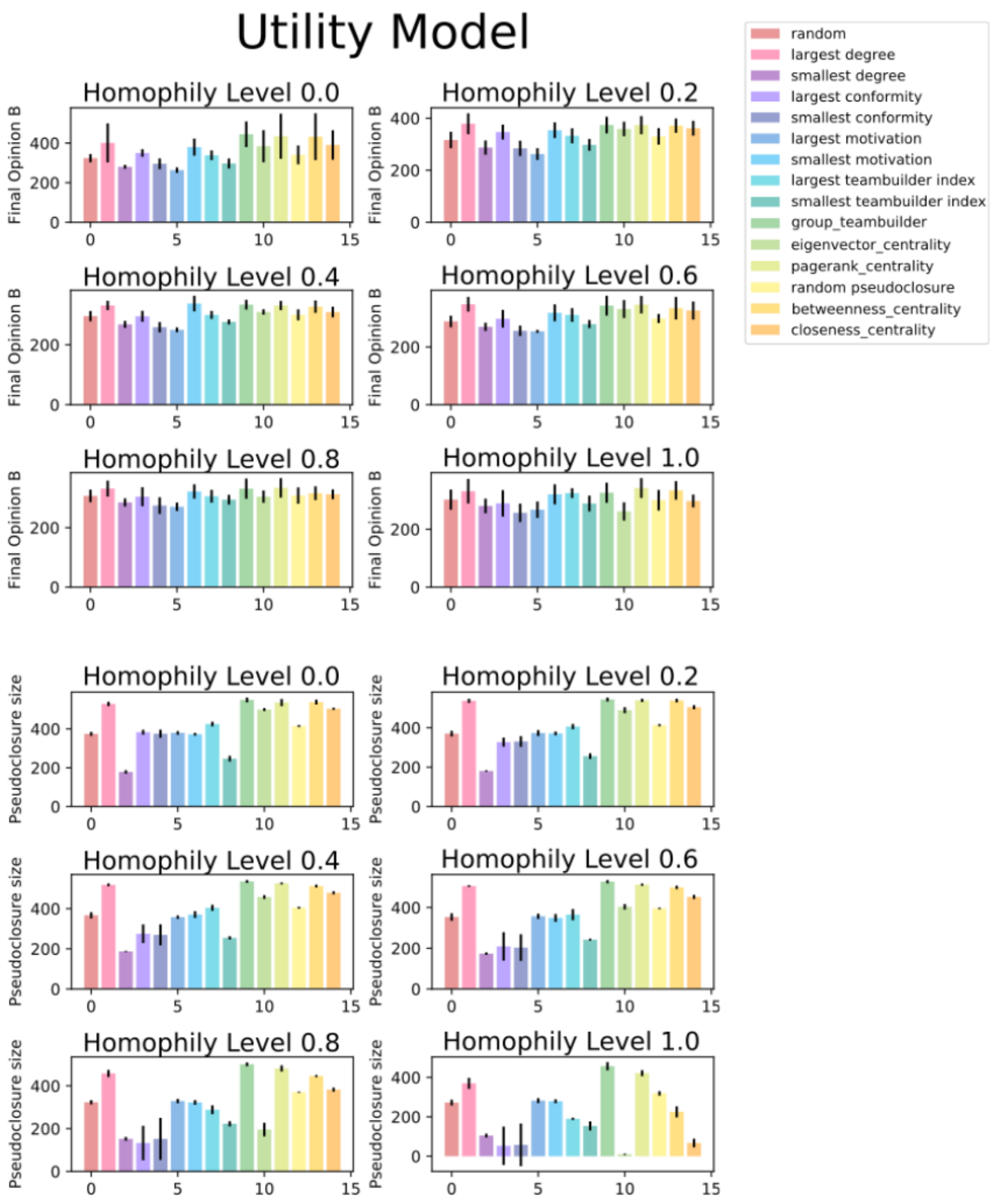


Figure 8.3: Results of the utility model of diffusion

pseudoclosure of a group if he/she has more than two friends in the group, and less than two enemies. The strategies that use this new pseudoclosure have a *mm* subindex ($e.g.pseudoclosure_{mm}$).

For this model 8.4 the difference of propagation can be seen from the smallest amounts of homophily level.

The size of the different pretopologies is also correlated to the final propagation in the mixed model, proving that our intuition in defining the $pseudoclosure_{mm}$ was also correct, but contrary to our beliefs neither the gti_{mm} nor the $random\_pseudoclosure_{mm}$ performed any better than regular *gti* or *pagerank* in identifying groups with large *pseudoclosure*$_{mm}$. The performance being quite similar for random networks, but getting increasingly worse as the homophily level augments.

It is worth noticing though that the *gti*$_{mm}$ heuristic was once again much more successful than *random_pseudoclosure*$_{mm}$ in identifying groups with large pseudoclosure.

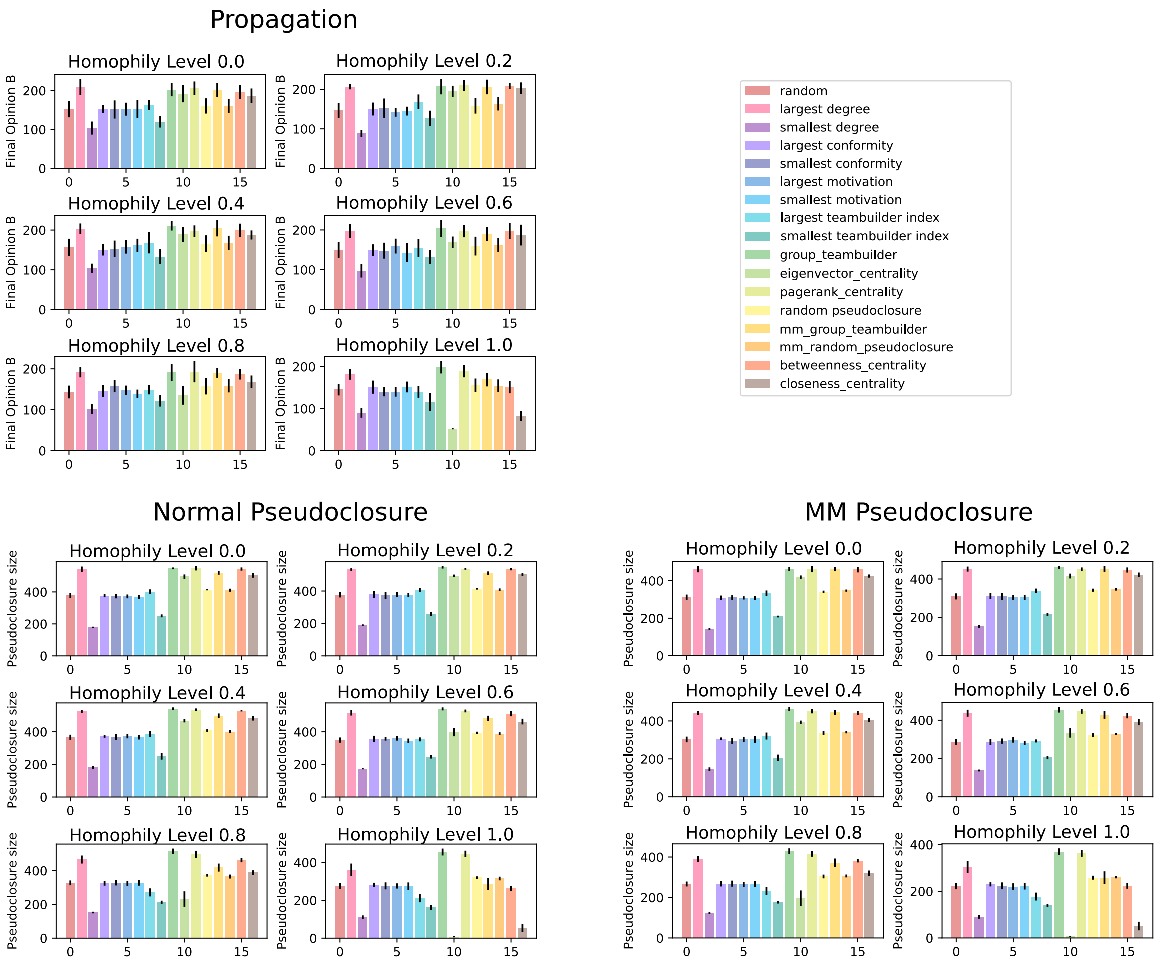


Figure 8.4: Results of the Mixed Model diffusion

8.2.3 Conclusion

Our most interesting results come from the heuristic based on the *teambuilder* notion, the *gti* or *group_teambuilder* strategy. This not only showed to be a better selection strategy for spreaders than all the network-based ones, but perhaps more importantly, seems to be a much better way to uncover sets with a large pseudoclosure than a simple random strategy, and this using a lot less resources. This last result is particularly important if we want to impose pretopology as a pertinent choice for structuring systems in a world where those systems and the data describing them are becoming ever larger.

8.3 Learning Propagation

Here we will take the same experimentation setting proposed in [36], in order to test our approach to learn a pretopological space. In this setting, the pseudoclosure function models the propagation of a forest fire inside of a grid. The set of cells of the grid is the universe set U of the pretopology. Each cell represents a section of the forest that can be either vulnerable, invulnerable or burning, and each cell has a set of neighborhoods that determines if it belongs to a pseudoclosure or not.

Although the authors of the paper talk about cells with a Moore neighborhood 8.3, this is different to the Moore neighborhood presented in the $\mathbb{Z}^2$ subsection. In that section we considered the Moore neighborhood as a single neighborhood, i.e. a prefilter basis with a single set of eight elements. Here, instead, it's considered as a prefilter basis with eight sets, each with one element. This gives eight different networks in our framework, and multiple different ways of combining them with a $DNF(.)$.

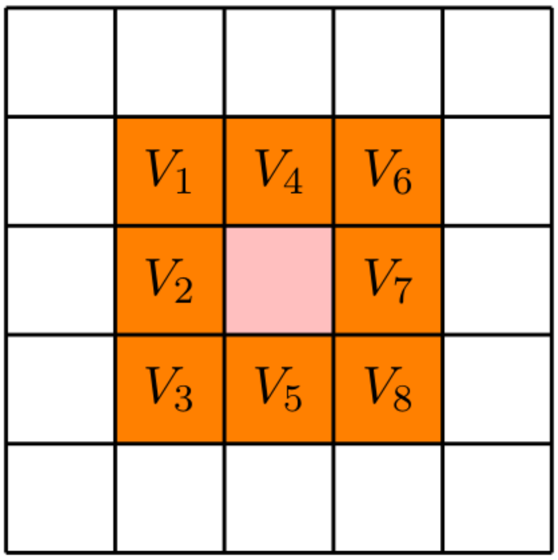


Figure 8.5: Different neighborhoods

Three different types of $DNF(.)$'s will be studied, each will produce a different type of pseudoclosure, and a different fire propagation:

- Simple = $V_3 \vee V_5 \vee V_6$
- Medium = $(V_3 \wedge V_5) \vee (V_4 \wedge V_7) \vee V_6$
- Hard = $V_2 \vee V_4 \vee (V_1 \wedge V_3) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6 \wedge V_7)$

Each $DNF(.)$ models a different propagation process influenced by the wind. For example, the simple $DNF(.)$ models a south-west wind influence that will make the fire propagate towards the up-right corner of the grid.

The model starts with a single cell burning, and the fire propagates by applying the pseudoclosure iteratively. The invulnerable cells have no neighborhoods other than their own singleton, so they cannot burn, nor propagate the fire. The set of burned cells at the moment when the fire stops expanding is an elementary closed set of the pretopology.

We will use six different environments to study the propagation; each will be a square grid with a different percentage (0%, 10%, 20%, 30%, 40%, 50%) of invulnerable cells chosen at random. Three sizes of grid will be tested: 15×15, 25×25 and 35×35.

For each $DNF(.)$ we will calculate the elementary closed set of 30% of the cells, choosing among vulnerable singletons, and we will pass it to the **BiggestSmallerClousure**(.) algorithm 6.3 in order to estimate the $DNF(.)$ and compare it to the original one.

8.3.1 Complexity

Before reviewing the results of the comparison, let's look at the complexity of our algorithm and check it for this particular case. For each closure C we need to see which elements $x \notin C$ belong to each network pseudoclosure $a_i(.)$. Now, if n is the number of cells per side of the grid (15, 25 or 35 in our example), we know that this can be done in $|\mathcal{C}| \cdot |\mathcal{G}| \cdot ((n^2)^2)$, where $\mathcal{C}$ is the set of closures, and $\mathcal{G}$ is the set of networks. This is just the complexity of one application of the pseudoclosure function for each closure7.1, since the number of different elements in the grid equals (n^2).

After we have done that, we will have the $forbidden\ conjunctions$. The rest of the algorithm 6.3 will be calculated in constant time for every size of the input, but it could be very large if the number of networks is. To get a superior bound for this second part we can take every possible conjunction $2^{|\mathcal{G}|}$, and for each of them compare it in terms of inclusion to every other $2^{|\mathcal{G}|}-1$ possible conjunction. This would give us an upper bound of $2^{2 \cdot |\mathcal{G}|}$ inclusion comparisons.

We can now estimate the time necessary for the whole algorithm as:

$$\begin{aligned} \tfrac{30}{100} \cdot n^2 \cdot 8 \cdot n^4 \cdot x_1 \ + \ 2^{16} \cdot x_2 &= \\ 2.4 \cdot n^6 \cdot x_1 \ + \ 2^{16} \cdot x_2 & \end{aligned}$$

where x_1 is the time needed for a multiplication and x_2 the time needed for an inclusion comparison. To test if our calculations were correct, we recorded the time taken to estimate the $DNF(.)$ for 40 different sides of grid, going from 10×10 to 50×50. We did this 10 times for each size, and we solved the following equation to get x_1 and x_2 :

$$\begin{bmatrix} 2.4 \cdot 10^6 & 2^{16} \\ 2.4 \cdot 11^6 & 2^{16} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{10} \\ a_{11} \end{bmatrix}$$

where a_{10} and a_{11} were the average times taken for the 10×10 and 11×11 grids, respectively.

A plot of the estimated function of time compared with the real time taken can be seen in figure 8.3.1. We can see that the growth pattern seems similar, and the real time never surpasses the estimated one.

It's important to realize that we are taking as input the side of the grid, which makes the number of elements (each cell of the grid) grow quadratically. For the more natural input of number of elements in the set, the algorithm would grow in n^3.

We finish pointing out that unless the number of networks is really big, the time taken for the second part is less than the time taken for the first part of the algorithm. For example, in this particular case x_1 and x_2 were on the same order of magnitude, and since we know that $10 > 2^3 \implies 10^6 > 2^{18} > 2^{16}$, then even for the smallest grid the time for the second part was less than the time for the first one, which was just the time of one pseudoclosure calculation. Loosely speaking we estimated the $DNF(.)$ in just the time necessary for two pseudoclosure calculations, when all previous algorithms required thousands of structural analysis calculations, which in turn required hundreds of pseudoclosure calculations.

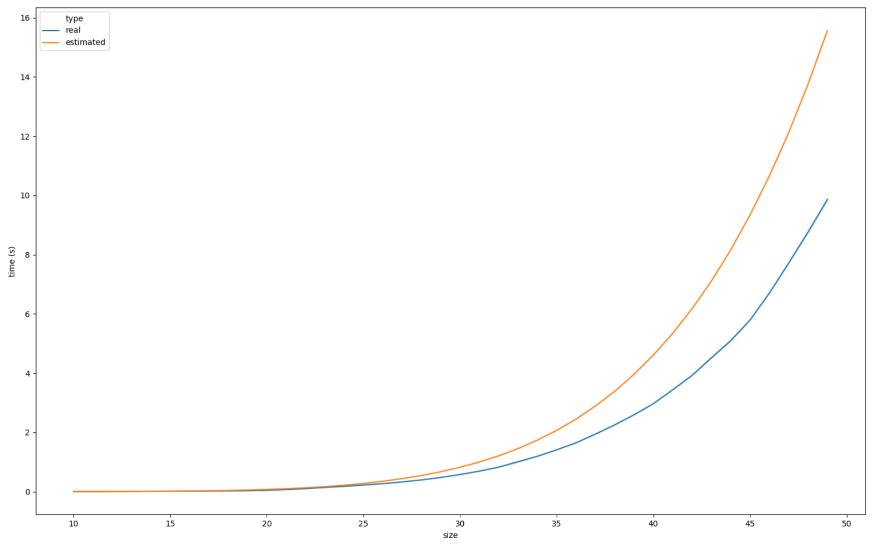


Figure 8.6: Time performance biggest smaller pseudoclosure

Average time comparison between the estimated and the real time of the algorithm over 10 runs, for different sizes of the grid. Errors are so small that are imperceptible.

8.3.2 Accuracy estimation

The original paper studies how close were the closures produced by the estimated $DNF(.)$ compared to the original ones. The same analysis wouldn't be of interest here, since our algorithm recovers the exact same closures every time. It's worth to notice that this perfect score in every setting wasn't achieved by any of the proposed algorithms, although the MI LPS was very close.

Instead, we will look at the $DNF(.)$ as a set of conjunctions, and we will study how close in terms of *precision* and *recall* are the estimated $DNF(.)$'s. We remind that the *precision* is defined as the ratio between the cardinality of the intersection of the sets and that of the estimated set, and the *recall* is defined as the ratio between the cardinality of the intersection of the sets and that of the original set. The results can be seen in tables 8.3.2 and 8.3.2. We can see that the differences in size and complexity of the $DNF(.)$ don't alter the results significantly, in opposition to the number of blocked cells, that looks quite determinant. Neither too few nor too many cells produce a perfect *recall*, while all the settings in between do that. The *precision*, on the other hand, is much lower, although better results are again produced around 30% of blocked cells. This low *precision* score seems correct, since our estimation is the fastest growing disjunctive normal form that will have the target closures, so its natural that for some settings the elimination of some conjunctions will produce the same closures at a slower pace.

We are not throwing away the results found in [36]. Our approach, although much more efficient in the case where a perfect solution to the problem exists, could potentially have a very poor performance (not in time but in prediction) where only approximative solutions exist.

blocked	15			25			35		
	simple	medium	hard	simple	medium	hard	simple	medium	hard
0	0.333	0.333	0.333	0.333	0.333	0.333	0.333	0.333	0.333
10	0.556	0.556	0.556	0.556	0.556	0.556	0.556	0.556	0.556
20	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563
30	0.628	0.226	0.532	0.628	0.305	0.532	0.628	0.368	0.532
40	0.499	0.499	0.499	0.499	0.499	0.499	0.499	0.499	0.499
50	0.414	0.414	0.414	0.414	0.414	0.414	0.414	0.414	0.414

Table 8.1: Average precision BiggestSmallerLSP
Average *precision* for 10 repetitions of the estimated $DNF(.)$ in terms of size of the grid, difficulty of the original $DNF(.)$ and percentage of blocked squares.

blocked	15			25			35		
	simple	medium	hard	simple	medium	hard	simple	medium	hard
0	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
20	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
30	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
40	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
50	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

Table 8.2: Average precission BiggestSmallerLSP
Average *recall* for 10 repetitions of the estimated $DNF(.)$ in terms of size of the grid, difficulty of the original $DNF(.)$ and percentage of blocked squares.

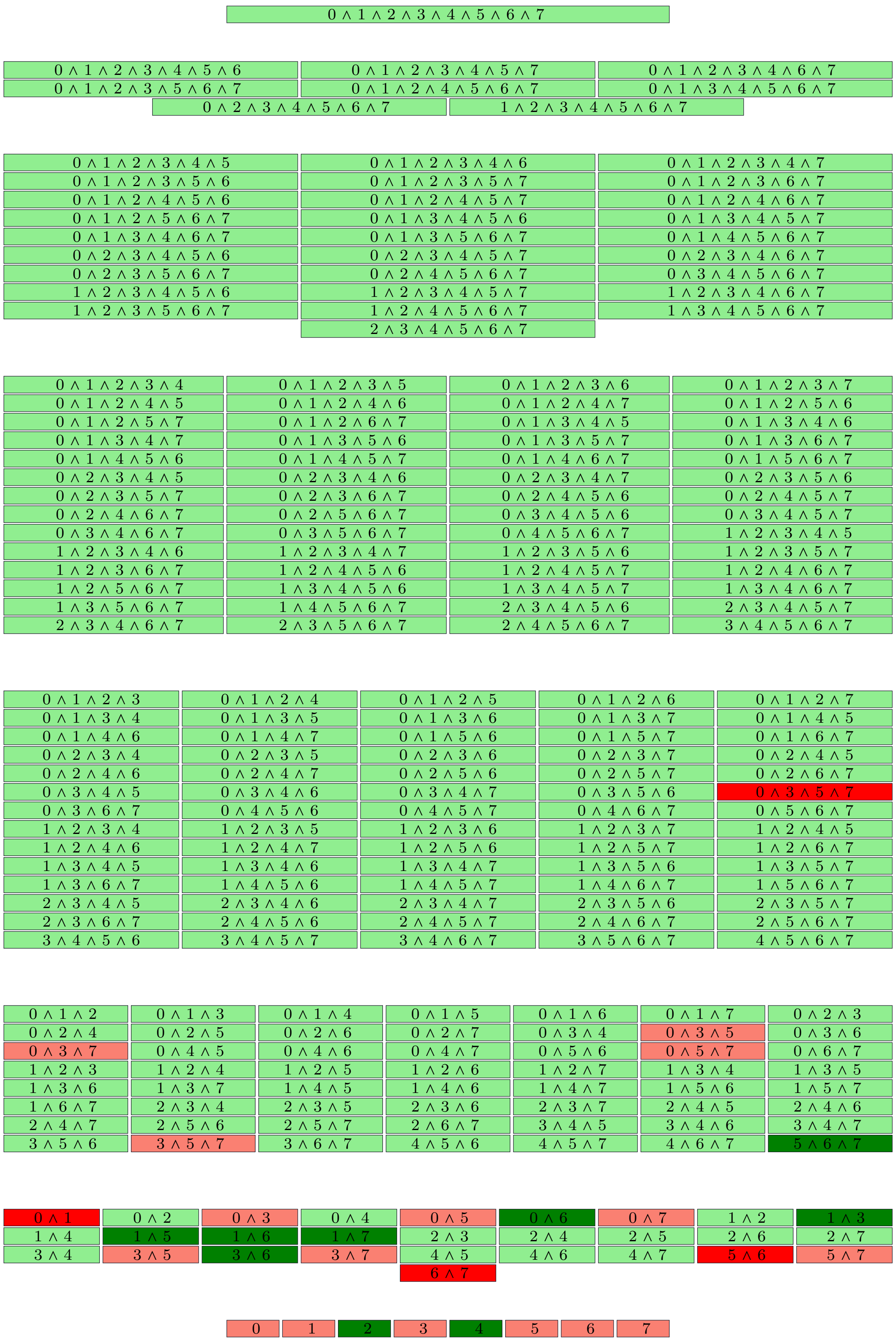


Figure 8.7: Example of DNF estimation

Example of closures for two different sizes of grid under the hard $DNF(.) = V_2 \vee V_4 \vee (V_1 \wedge V_3) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6 \wedge V_7)$. On red the combinations of neighborhoods that didn't propagate; on salmon those run out because they are more general than the red ones; on dark green those kept; on light green those run out because they are more specific than those already selected. The forbidden conjunctions are: $(V_0 \wedge V_1)$, $(V_5 \wedge V_6)$, $(V_6 \wedge V_7)$ and $(V_0 \wedge V_3 \wedge V_5 \wedge V_7)$, and the estimated $DNF(.)$ is $V_2 \vee V_4 \vee (V_0 \wedge V_6) \vee (V_1 \wedge V_3) \vee (V_1 \wedge V_5) \vee (V_1 \wedge V_6) \vee (V_1 \wedge V_7) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6 \wedge V_7)$

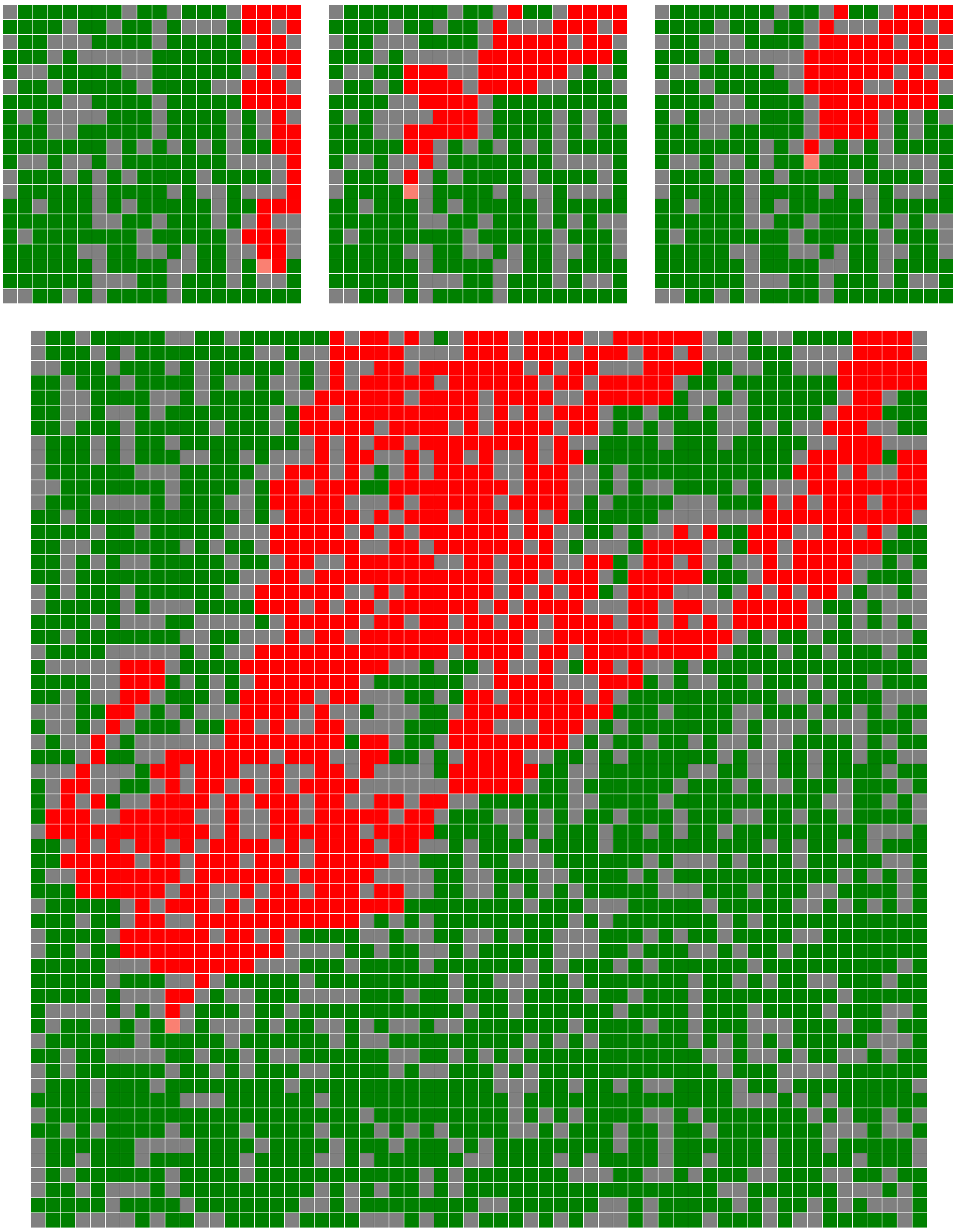


Figure 8.8: Cluster Pseudoclosure

Example of closures for two different sizes of grid under the hard $DNF(.) = V_2 \vee V_4 \vee (V_1 \wedge V_3) \vee (V_3 \wedge V_6) \vee (V_5 \wedge V_6 \wedge V_7)$. On top, three closures for a simulation with a 20×20 square grid. On the bottom one closure for a simulation with a 60×60 square grid.

Chapter 9

Clustering

This chapter explains how some of the concepts that have been introduced so far can be used to find clusters inside of a space. There is not a precise definition of what a cluster is, but the general idea is that of forming groups inside of a set, such that elements in a same group are similar among them, and dissimilar from the rest. We compare our ideas with some of the classical clusterization algorithms of points in $\mathbb{R}^2$, and we show that our algorithm performs at pair with the state of the art, but is much more general, being easily adapted to non-metric spaces.

9.1 Previous Pretopological works on Clusterization

The concepts of pretopology have already been used for the study of clusterization in a series of works. Although these works differ in strategy and applications, they all have in common the use of pretopology as a way of improving the *K-means* algorithm.

Although the term *K-means* was introduced in [98], what has come to be known as [?] nowadays is the following algorithm presented in [96]:

- We define the number k of clusters we wish to identify.
- We select k random points P_i, $i \in \{1, 2, \ldots, k\}$.
- We add each element of our set to the cluster where the closest P_i belongs.
- We calculate the centroids of each cluster.
- We repeat the process with the centroids as the new set of P_i points.

This is an approximation algorithm for the original NP-hard problem, where given a set S of points in $\mathbb{R}^d$, we need to find k points called *centers*, such that the sum of the squared Euclidean distance of each point in S to its closest center should be minimal. The algorithm converges usually quite fast to a local minimum.

Although the algorithm has proven to perform quite well in many situations, it has a certain number of drawbacks. First, it is a random algorithm, sensible to the choice of the initial set of points P_i. This is usually dealt with by making multiple runs of the algorithm and selecting the one with the best performance, but this increases the calculations, and as with any random algorithm we may still make a poor selection. A second problem is that the number of clusters should be known in advanced, something that is usually not the case in practice.

The following pretopological algorithm was proposed to overcome those problems.

9.1.1 MCPR

In [87, 88], Le *et al.* introduced the **MCPR** algorithm. They proposed to use the structural analysis algorithm presented in5.1 before using *K-means*, so that the number of clusters was equal to the number of *minimal closed sets*. They also defined a distance inside a pretopological space, used it to determine the most densely connected element inside each *minimal closed sets*, and selected the element as the initial center of the cluster. Finally, the same distance was used to decide the closest center to each element of the space, allowing to use *K-means* in non-metric pretopological spaces.

Recently, in [32], Bui *et al.* used these notions to clusterize a corpus of documents. They used Latent Dirichlet Allocation (LDA) to identify the topics of each document, and built two kinds of document relations: one using the mayor topic, the other using the Hellinger distance between the topic vectors. Eventually they defined a weak pseudoclosure function from those two relations, and applied the **MCPR** algorithm. Their results were found to be competitive with the state of the art.

Although these have been great improvements to the standard $K-means$ algorithm, some drawbacks still remain: one is that $K - means$ is a partition algorithm (i.e. every set ends up in a cluster), and this makes impossible the detection of noise in the original set; another is that clusters have more or less equal size.

A final issue concerning the **MCPR** is it computational cost. Indeed, Levorato [90] estimates the complexity of this algorithm for a worst case scenario as $(O)(n^5)$ in terms of the number of pseudoclosure operations necessary. Since we saw that our framework calculates the pseudoclosure in $(O)(n^2)$, with n being the number of elements in the space, we get a total complexity of $(O)(n^7)$ in terms of the size of the space. Even if polynomial, this becomes prohibitively expensive for sets relatively small.

In the next section we present a new pretopological method of clusterization that solves the aforementioned problems, all while keeping the possibility of finding clusters in non-metric spaces.

9.2 Prepoclusters

In this section we clusterize using the $degree-n$ generalizations of the structural analysis introduced in sections *structural analysis* 5.1 and *quasihierarchy* 5.3. The idea is quite simple, we create a pretopological space, we calculate the quasihierarchy of *elementary closed sets of degree-n*, and we identify the biggest elementary closed sets with the clusters. These are the elements that are on top of the hierarchy.

Using the $degree - n$ generalizations we are able to formalize the following simple notion: **an element x joins a group if the elements that are closer to it inside the group, are not much closer among them, than what they are to x.** The idea is to make a cluster grow adding elements that will not alter the density too much. This prevents two clusters that have an element between them to be joined into one cluster. That intermediary element will probably join both clusters, but this will not make a point in cluster A to be sufficiently attracted to cluster B as a whole. The addition of an element to a group is a property of the group, which is exactly what pretopology can model very well.

This is where the definition of *degree-n* elementary sets becomes important, since we need to start the cluster with a set that is sufficiently big to attract some elements, but only if they are well connected to the group as a whole.

To formalize the notion just mentioned we need to define what we mean by "being close". We figured this definition was dependent on the density of the points, so we devised an heuristic to quantify the density. For this, we defined a variable *square_length* and a variable *real_points*:

- *square_length* is the length of a grid cell, if we divided the smallest rectangle that covers all points, into a grid with *points* cells; *points* being the number of points in the dataset. More specifically:
  - We take the difference Δx between the point that is more to the right, and the one that is more to the left.
  - We take the difference Δy between the highest, and the lowest point.
  - We take the area of the rectangle $\Delta x \times \Delta y$, and we divide it by *points*, the number of points in the dataset.
  - We have now the area corresponding to each point, so we take the square root of that value to get the *square_length*
- To calculate *real_points* we consider that all points that are inside a same square of length *square_length* are a unique point, and *real_points* is the number of those points.

What we are doing is basically estimate how many cells of the grid would be occupied, if we were to define a grid over the dataset.

Having this in mind, let us pass review to the exact parameters that are needed for the algorithm:

- The degree n of the initial sets.
- The *radius* of the balls to make the geometric network over which we will build the pretopology. *I.e*, pairs of points that are closer than *radius* are connected in the network, the rest are not. The strength of the link between x and y will be equal to $1 - \frac{distance_{xy}}{radius}$, to that closer points will have bigger strength.
- The *threshold* θ_N associated to the network in our framework.
- The threshold θ_Q for the strength of the links in the quasihierarchy 5.3. *I.e*, after calculating the strength of the links between the different quasiclosures, those that are stronger than θ_Q are considered connected, the others are not.

These parameters needed to be chosen manually, and it may seem difficult to do so in a way that proves to be useful, but we tested the same set of parameters over six different datasets with great results, and we only needed a slight modification to make it work on the seventh dataset.

We used $degree = 4$ and $\theta_Q = 0.5$ for all the datasets.

For the six datasets shown in figure 9.1 we used $radius = 2 \times square$ and $theta_N = \frac{points}{3 \times real\_points}$. These are parameters that were found by some trial and error, guided by the density of the points. The figure shows the results of the pretopolclusters compared with a series of classical clusterization algorithms. These are the data used by scikit-learn, the standard package of machine learning in python, for their comparison of the algorithms. We can see that the pretopoclusters always find the clusters we would hope for it to find; an exploit only met by DBSCAN. We can also see how those are the only two algorithms capable of identifying noise.

For a second data set we used $radius = 2 \times square$ and $theta_N = points/real\_points$. Figure 9.2 shows the results for the *K-means* and *Agglomerative*algorithm. We can see that although the parameters where chosen wisely (six clusters were supposed to be found), they perform very poorly because of the multiple drawbacks we have mentioned.

Figure 9.3 shows the results for the *HDBSCAND* and *Pretopoloclusters* algorithm. HDBSCAN is a recent algorithm that "extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters". This algorithm has shown even better performance than DBSCAN in situations where clusters have many scarce points between them. We can see that our algorithm finds almost the exact same clusters. It should be mentioned that for this configuration, those sets that did not grow to twice their size were considered noise.

The difference in parametrization between the first group of six datasets and the second is mostly due to the variance in density over different zones that is found in the second dataset. Future works may include different networks according to different densities to better account for this.

9.2.1 Final remarks

We have seen how the concepts of pretopology allow us to find clusters in a way that is at pair with the state of the art. On the other hand, this is but the first work in a direction that looks very promising to us.

A particularly important aspect of the algorithm proposed is its possibility to be extended to $non-V$ pretopologies. Our example is built on a pretopological space of type V, but the fundamental property for the structuring process is not met, notably that the points in the intersection of a pair of sets, belong only to sets that are perfectly contained in that intersection. This is the property that fails in a general pretopological space preventing its structural analysis, so this extension solves the problem.

We should also notice that our approach is fundamentally different from other hierarchical approaches to clustering, in the sense that those algorithms start from singleton clusters and join different clusters one by one, until the whole space is one big cluster. We then need to decide a *cut* in the hierarchy, that is, at what point in the process of building the hierarchy we will stop joining clusters together and consider that we have found the clusters we were looking for. In the case of pretopoclusters there is no need to define such a cut zone, since the clusters emerge naturally as the biggest elementary quasiclosed sets. Also, as with *DBSCAN*, our approach allows the identification of noise.

A final characteristic is that our algorithm can find elements belonging to two clusters, an idea already present in the recent literature about network community detection [61]

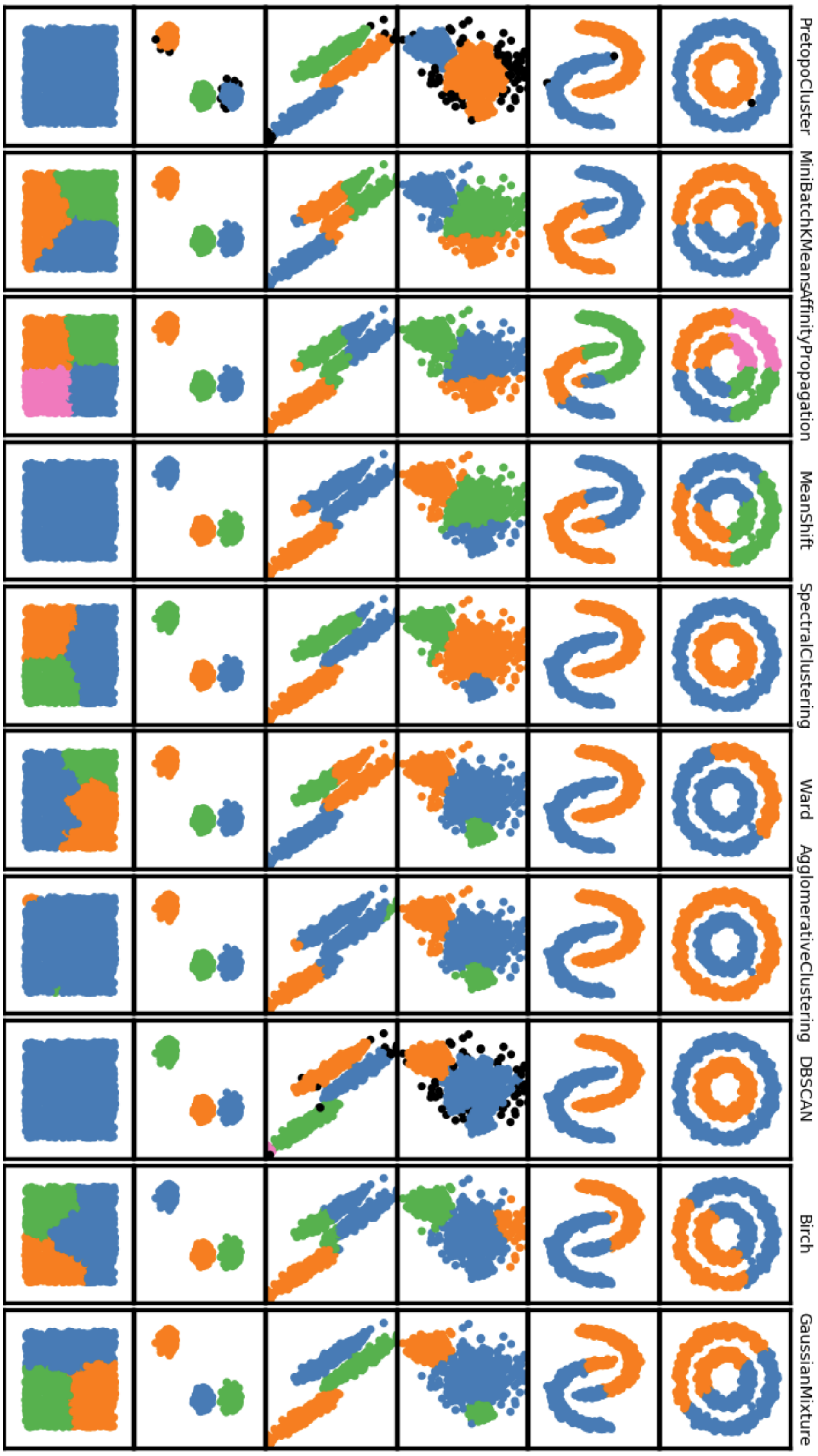
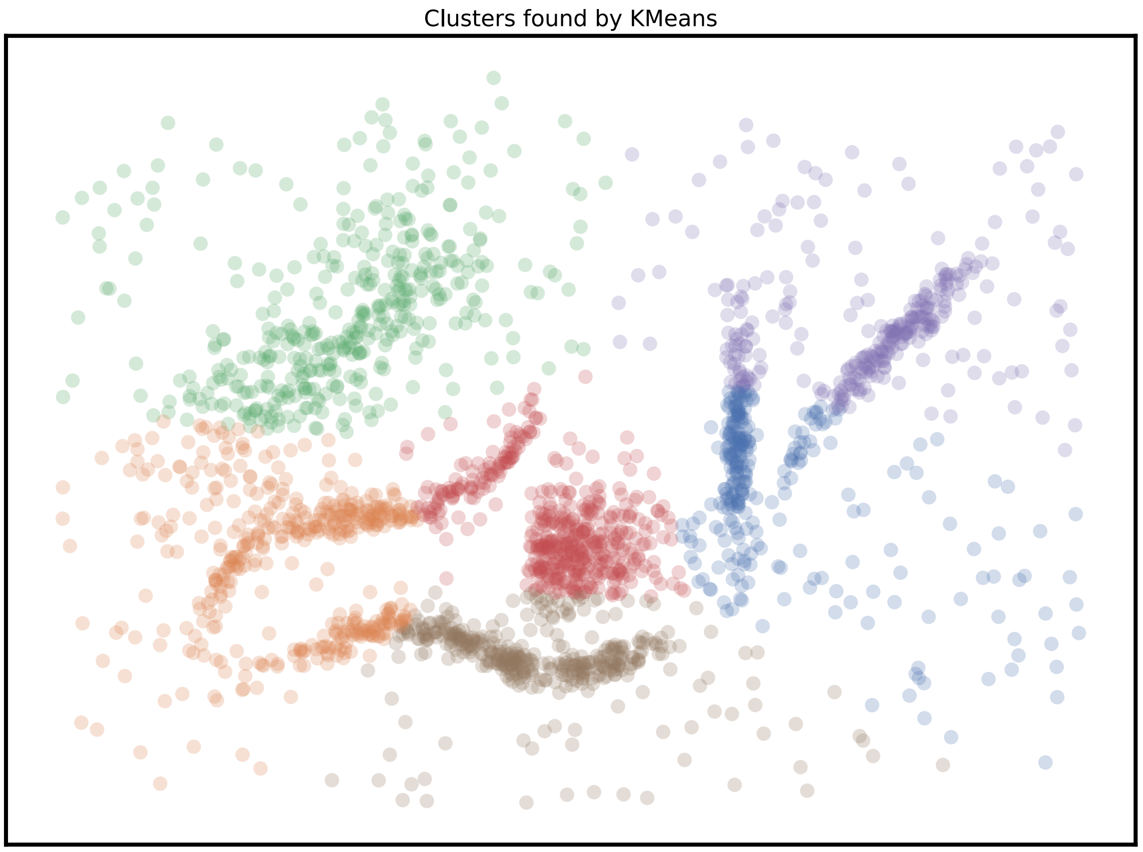


Figure 9.1: Clustering Results Comparison



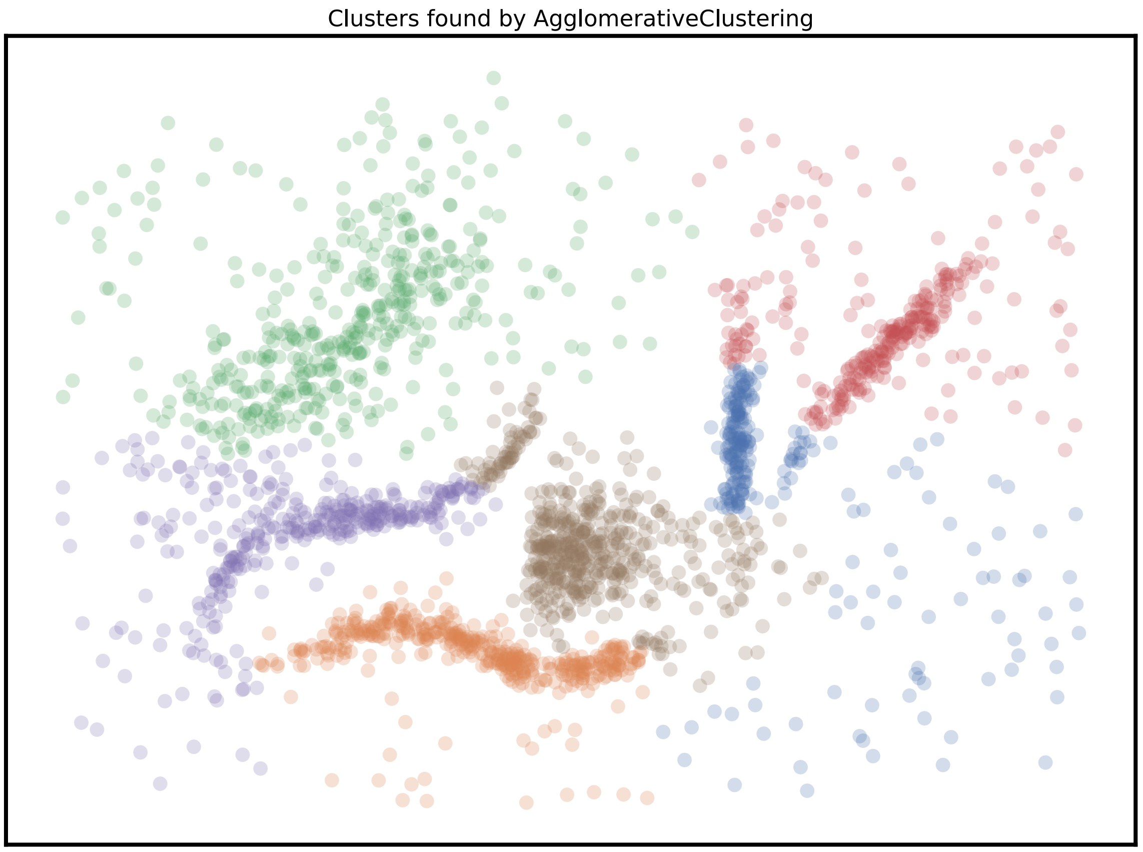
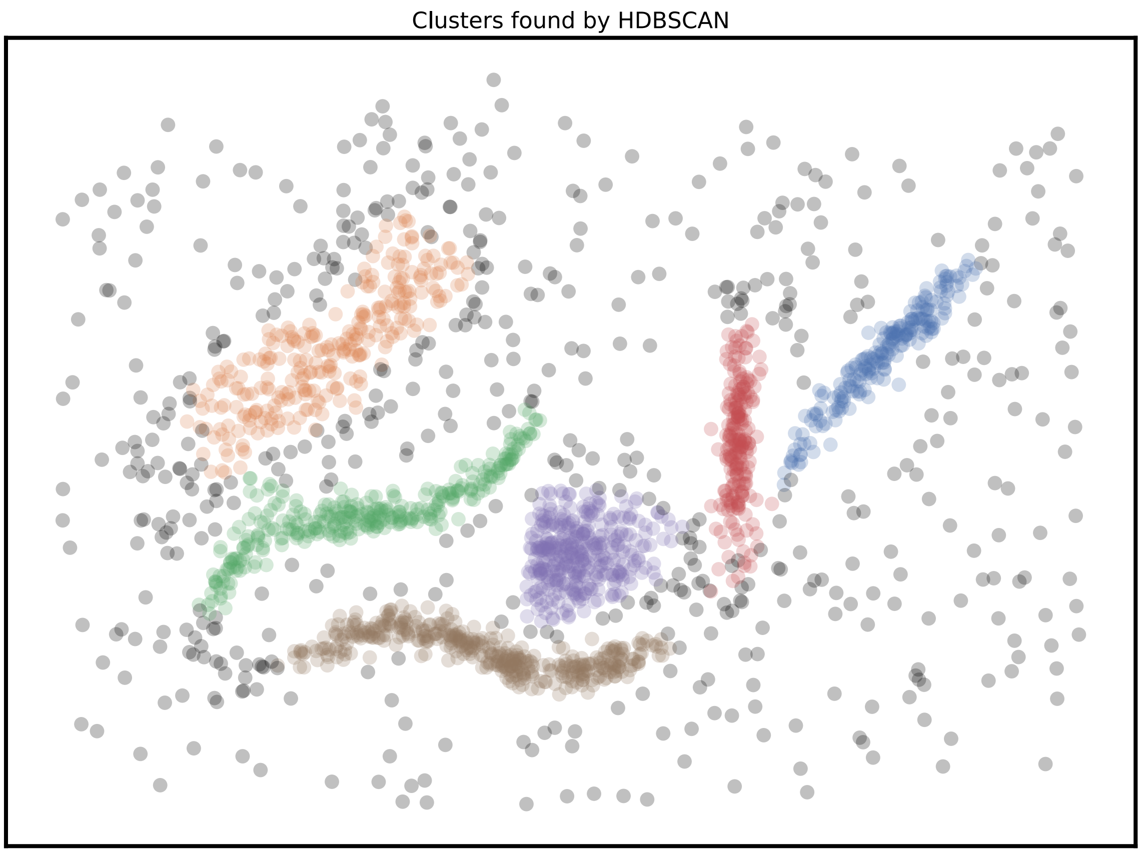


Figure 9.2: Cluster K-Means and Agglomerative



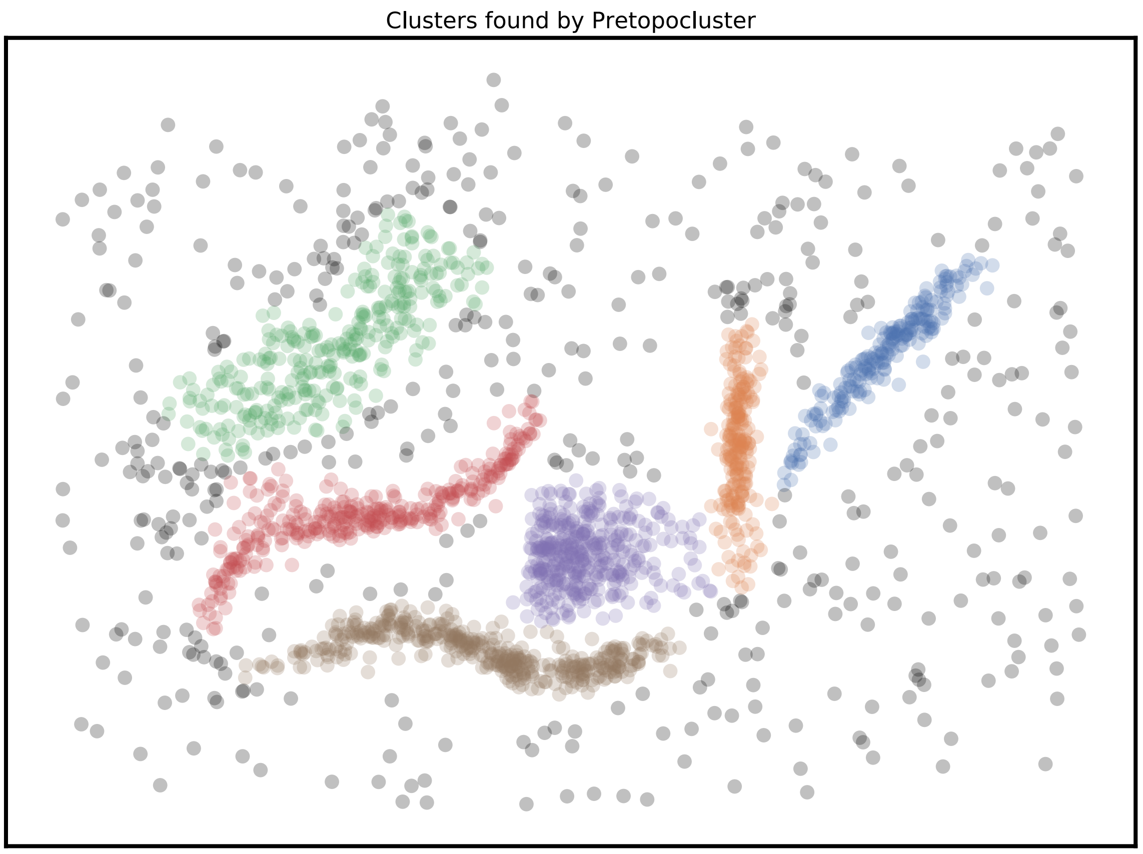


Figure 9.3: Cluster HDBSCAND and Pretopocluster

Conclusion

Chapter 10

Conclusion

We have seen throughout the thesis many examples of how pretopology could be an appropriate way to model the structure of some real world phenomena, capable of modeling phenomena that could not be represented with network theory without loosing information.

We have made an effort to show that pretopology needs not be just a modeling tool, but it can actually become a practical way to study the systems it models. We proved, for example, that some pretopological measures of centrality actually work best in certain context.

A great effort was put into devising an economical and practical way to treat a pretopological space; one that would allow us to store a space and implement the classical algorithms without having to change the way we conceive a pretopology, as a set theoretic framework. It was shown through many examples how general this framework is, and how practical it was to study the complexity of the existent algorithms and to connect our theory with some very active fields of research, such as network theory and ILP. It was also the framework that permitted us to implement in a practical fashion a pretopological Python library.

Our most interesting result is probably the one concerning the quasihierarchies. Not only we devised a way to extend the structural analysis to every kind of pretopological space, but we showed how effective it was dealing with clustering problems.

It is also in that area that we see the biggest prospects. When we used quasihierarchies to find clusters, a fair amount of manual parametrization was made. The next logical step in the development of the theory will be to put the notions of quasihierarchy and LPS to work together, in order to get a semi-supervised way to learn to recognize clusters.

It seems to us particularly promising the idea of using a $non-V$ pretopological space to model the semantic relations between words and groups, and to use the quasihierarchy concept along with the LPS algorithms to uncover the topics of a document.

Other possible prospects are the following:

- Explore the generalizations mentioned in section three: dynamic petopology, random pretopology, weighted pretopology.
- Improve an add new functions to the library. A particularly urgent change concerns the store of sparse matrices. Indeed, we are storing the whole adjacency matrix for every network, and we have seen that the formalization of pretopological spaces described by some pseudoclosures needs a huge number of networks.
- Better explore the connections to ILP and network algorithms. Variations of the algorithms related to the minimum domination set may be particularly helpful in finding sets with large pseudoclosures.

Appendices

Appendix A

Python Library

class Prenetwork
Description: A prenetwork is the combination of a network -descrived as its adjacency matrix- and a list list of tuples ($threshold, weight$) associated to it.

- Attributes:
  - network:
    The adjacency matrix that describes this network.
  - weights:
    A list of weights associated to each network
  - thresholds:
    A list of thresholds associated to each network

Comments: A prenetwork is a way of storing multiple networks in one single adjacency matrix. One prenetwork represents each of the networks obtained by multiplying the adjacency matrix by one of the weights, and the corresponding threshold associated to it.

This will also allow some improvements on the calculation of the pseudolosure , since the matrix multiplication will be done only once, and the resulting vector is the one that will be multiplied by the different weights and compared to the different thresholds.

One example of the utility of this is given by the use of a network with 1 and -1 as weights, and θ and $-\theta$ as thresholds. This allow us to model the membership to a pseudoclosure and its negation in one network.

Example: A Prenetwork with adjacency matrix A and list of tuples [(1, 1),(2, 1),(1,2)], is equivalent to the pretopology described by three networks with adjacency matrices $\mathbf{A}$, $\mathbf{A}$ and $\mathbf{2A}$, and three thresholds $\theta_1 = 1$, $\theta_2 = 2$ and $\theta_3 = 1$. This not only allows us to stock multiple networks saving a lot of space, but also the pseudoclosure can be calculated much faster, since the matrix multiplication will be done only once, and the resulting vector is the one that will be multiplied by the different weights and compared to the different thresholds.

class PretopologicalSpace

Description: This is the class that has all the basic functionalities of a general pretopological space defined under our network.

- Attributes:
  - prenetworks:
    A list of the Prenetworks (*c.r.* **class Prenetwork**) that define a space.
  - prenetwork_indices:
    A list of the prenetwork indexes associated to each network. e.g. If the first prenetwork represents five different networks, then the list will start with five zeros.
  - thresholds:
    A list of the thresholds associated to each network.
  - weights: A list of the weights associated to each network..
  - dnf:
    The $DNF(.)$ that characterizes the pretopological space, represented as a list of lists. Each of the nested lists contains the indices of the networks that belong to a same conjunction.
  - size:
    Returns the number of elements of the space.
- Methods:
  - pseudoclosure(set):
    Returns a set with the elements of the pseudoclosure of *set.*
  - add_ prenetworks($prenetworks$):
    Adds the elements of *prenetworks* to the list of prenetworks of the space. It also updates the *network_indices*, *thresholds* and *weights* attributes of the space.
  - add_ conjunction(conjunction):
    Adds a conjunction, represented by a list of indices, to *dnf.*

Comments: Every time we talk about a *set*, we are talking about its membership numpy array.

class PretopologicalSpaceEnv(PretopologicalSpace)
Description: The class extends the **PretopologicalSpace** class so it can have an environment associated.

- Attributes:
    - environment: A numpy matrix where each row represents an element of the space, and each column is the value of a characteristic of the element. The environment could be associated to a population of agents.
    - attribute_labels: A list of strings specifying the name of the attributes represented by the columns.

Comments: The purpose of this class is build pretopologies on sets with more complex elements. This will allow to define different rules for the pseudoclosure based on the different characteristics of the elements.

class PretopologicalSpaceGrid(PretopologicalSpaceEnv)
Description: The class extends the **PretopologicalSpaceEnv** class so the environment associated to it can be a interpreted as a grid.

- Attributes:
  - prenetworks: a list
  - network_ indices:
  - thresholds:
  - dnf:
  - size:
- Methods:
  - create_n0(): Creates a network where each cell is connected to the one up and to the left.
  - create_n1(): Creates a network where each cell is connected to the one to the left.
  - create_n2(): Creates a network where each cell is connected to the one down and to the left.
  - create_n3(): Creates a network where each cell is connected to the one right up.
  - create_n4(): Creates a network where each cell is connected to the one right down.
  - create_n5(): Creates a network where each cell is connected to the one up and to the right.
  - create_n6(): Creates a network where each cell is connected to the one to the right.
  - create_n7(): Creates a network where each cell is connected to the one down and to the right.

Comments: The names of the neighborhoods are the same used in section 8.2.

module Metrics
Description: This is the module with functions necessary to calculate the biggest pseudoclosure and the team builder index.

- teambuilder(*set, size*):
  Returns the set of teambuilder indices for the elements of the set as descrived in the section dedicated to the teambuilder notion (*i.e.* 4.2).
  For the moment the method can only be used with pretopologies descrived by a $DNF(.)$ with a single conjonction with one clause.

- biggest_pseudoclosure_greedy_teambuilder(*set, size*):
  Returns the set of size=*size* with the biggest pseudoclosure, estimated using the method descrived in the section dedicated to the teambuilder notion (*i.e.* 4.2).

- biggest_pseudoclosure_ilp(*set, size, solver="ECOS BB"*):
  Returns the set of size=*size* with the biggest pseudoclosure, estimated using the method descrived in the section dedicated to ILP (*i.e.* 4.1).
  This method can only be used with pretopologies descrived by a $DNF(.)$ with only one conjonction.
  For the moment the method only works with the solver ECOS BB through the python package for convex linear programming cvxpy citation

module Closures
Description: This is the module with functions necessary to calculate the closures of degree n of a list of sets.

Bibliography

[1] *Basics of Pretopology.*

[2] M. Ahat, S. B. Amor, M. Bui, S. Jhean-Larose, and G. Denhiere. Document Classification with LSA and Pretopology. page 21.

[3] E. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing, and A. X. Zheng, editors. *Statistical Network Analysis: Models, Issues, and New Directions*, volume 4503 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[4] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, Jan. 2002.

[5] A. Aleta and Y. Moreno. Multilayer Networks in a Nutshell. *arXiv:1804.03488 [cond-mat, physics:physics]*, Apr. 2018. arXiv: 1804.03488.

[6] J. G. Aluja and A. M. G. Lafuente. *Towards an Advanced Modelling of Complex Economic Phenomena*, volume 276 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[7] J.-P. Auray, S. Bonnevay, M. Bui, G. Duru, and M. Lamure. Prétopologie et applications : un état de l'art. page 18.

[8] J.-P. Auray, G. Duru, and M. Mougeot. A pre-topological analysis of the input-output model. *Economics Letters*, 2(4):343–347, Jan. 1979.

[9] R. Axelrod. Chapter 33 Agent-based Modeling as a Bridge Between Disciplines. In *Handbook of Computational Economics*, volume 2, pages 1565–1584. Elsevier, 2006.

[10] R. Axtell. WHY AGENTS? ON THE VARIED MOTIVATIONS FOR AGENT COMPUTING IN THE SOCIAL SCIENCES. page 23.

[11] A.-L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. 286:4, 1999.

[12] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, Cambridge, United Kingdom, 2016. OCLC: ocn910772793.

[13] M. Barlet, D. Blanchet, and T. Le Barbanchon. Microsimulation et modèles d'agents : une approche alternative pour l'évaluation des politiques d'emploi. *Economie et statistique*, 429(1):51–76, 2009.

[14] J. Barthelemy and P. L. Toint. Synthetic Population Generation Without a Sample. *Transportation Science*, 47(2):266–279, May 2013.

[15] C. Basileu. Modélisation structurelle des réseaux sociaux : application à un système d'aide à la décision en cas de crise sanitaire. page 206.

[16] C. Basileu, S. B. Amor, M. Bui, and M. Lamure. Prétopologie stochastique et réseaux complexes. page 66.

[17] A. Bazin. A depth-first search algorithm for computing pseudo-closed sets. *Discrete Applied Mathematics*, 249:28–35, Nov. 2018.

[18] R. J. Beckman, K. A. Baggerly, and M. D. McKay. Creating synthetic baseline populations. *Transportation Research Part A: Policy and Practice*, 30(6):415–429, Nov. 1996.

[19] Y. Ben-Zion, Y. Cohen, and N. M. Shnerb. Modeling epidemics dynamics on heterogenous networks. *Journal of Theoretical Biology*, 264(2):197–204, May 2010.

[20] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1):92–128, Feb. 2005.

[21] B. Bollobás. *Modern graph theory.* Number 184 in Graduate texts in mathematics. Springer, New York, 1998.

[22] S. Bonnevay. Pretopological Operators for Gray-Level Image Analysis. page 17.

[23] S. Bonnevay, M. Lamure, C. Largeron-Leteno, and N. Nicoloyannis. A pretopological approach for structuring data in non-metric spaces. *Electronic Notes in Discrete Mathematics*, 2:1–9, Apr. 1999.

[24] S. P. Borgatti and M. G. Everett. Notions of Position in Social Network Analysis. *Sociological Methodology*, 22:1, 1992.

[25] S. P. Borgatti and M. G. Everett. A Graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, Oct. 2006.

[26] M. Bouden, B. Moulin, P. Gosselin, W. avenue, and P. Gosselin. Epidemic Propagation of West Nile Virus Using a Multi-Agent Geo-Simulation under Various Short-term Climate Scenarios. page 8, 2008.

[27] L. Boudin and F. Salvarani. A kinetic approach to the study of opinion formation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(3):507–522, May 2009.

[28] M. Brissaud, J. P. Auray, G. Duru, and M. Lamure. Eléments de prétopologie généralisée. page 32.

[29] Q. V. Bui. *Pretopology and Topic Modeling for Complex Systems Analysis: Application on Document Classification and Complex Network Analysis.* PhD thesis.

[30] Q. V. Bui, S. B. Amor, and M. Bui. Stochastic Pretopology as a tool for Complex Networks Analysis. page 21.

[31] Q. V. Bui, S. B. Amor, and M. Bui. Stochastic Pretopology as a tool for Complex Networks Analysis. page 21.

[32] Q. V. Bui, K. Sayadi, and M. Bui. A multi-criteria document clustering method based on topic modeling and pseudoclosure function. In *Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015*, pages 1–8, Hue City, Viet Nam, 2015. ACM Press.

[33] Q. V. Bui, K. Sayadi, and M. Bui. A multi-criteria document clustering method based on topic modeling and pseudoclosure function. In *Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015*, pages 1–8, Hue City, Viet Nam, 2015. ACM Press.

[34] Q. V. Bui, K. Sayadi, and M. Bui. A multi-criteria document clustering method based on topic modeling and pseudoclosure function. In *Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015*, pages 1–8, Hue City, Viet Nam, 2015. ACM Press.

[35] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, Apr. 2010.

[36] G. Caillaut and G. Cleuziou. Learning Pretopological Spaces to Model Complex Propagation Phenomena: A Multiple Instance Learning Approach Based on a Logical Modeling. *arXiv:1805.01278 [cs, stat]*, May 2018. arXiv: 1805.01278.

[37] M. Chellali, O. Favaron, T. W. Haynes, S. T. Hedetniemi, and A. McRae. Independent [1, k]-sets in graphs. *AUSTRALAS. J. COMBIN.*, page 13, 2014.

[38] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence Maximization in Social Networks When Negative Opinions May Emerge and Propagate. In B. Liu, H. Liu, C. Clifton, T. Washio, and C. Kamath, editors, *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 379–390. Society for Industrial and Applied Mathematics, Philadelphia, PA, Apr. 2011.

[39] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703, Nov. 2009.

[40] P. Clayton and P. Davies. *The re-emergence of emergence: the emergentist hypothesis from science to religion.* Oxford University Press, Oxford; New York, 2006. OCLC: 764564268.

[41] G. Cleuziou and G. Dias. Learning Pretopological Spaces for Lexical Taxonomy Acquisition. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, and C. Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 9285, pages 493–508. Springer International Publishing, Cham, 2015.

[42] R. Cohen, S. Havlin, and D. ben Avraham. Efficient Immunization Strategies for Computer Networks and Populations. *Physical Review Letters*, 91(24), Dec. 2003.

[43] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani. The Modeling of Global Epidemics: Stochastic Dynamics and Predictability. *Bulletin of Mathematical Biology*, 68(8):1893–1921, Nov. 2006.

[44] M. Dalud-Vincent. Une autre manière de modéliser les réseaux sociaux. Applications à l'étude de co-publications. *Nouvelles perspectives en sciences sociales*, 12(2):41, 2017.

[45] M. Dalud-Vincent, M. Brissaud, and M. Lamure. Closed Sets and Closures in Pretopology. page 12.

[46] M. Dalud-Vincent, M. Brissaud, and M. Lamure. Pretopology, Matroids and Hypergraphs. page 14.

[47] M. Dalud-Vincent, M. Forsé, and J.-P. Auray. An algorithm for finding the structure of social groups. *Social Networks*, 16(2):137–162, Apr. 1994.

[48] M. Danisch. Limass: Linear algorithms for massive real-world graphs. `https://sites.google.com/view/limass/accueil?authuser=0`. Accessed: 2019-01-30.

[49] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas. Mathematical Formulation of Multilayer Networks. *Physical Review X*, 3(4), Dec. 2013.

[50] B. L. Dietrich. Matroids and antimatroids—a survey. *Discrete Mathematics*, 78(3):223–237, 1989.

[51] V. Dignum. Social agents: bridging simulation and engineering. *Communications of the ACM*, 60(11):32–34, Oct. 2017.

[52] I. Douven and A. Riegler. Extending the Hegselmann-Krause Model I. *Logic Journal of IGPL*, 18(2):323–335, Apr. 2010.

[53] G. Eliasson, editor. *A micro-to-macro model of the Swedish economy: papers on the Swedish model from the Symposium on Micro Simulation Methods in Stockholm, September 19-22,1977.* Number 1978,1 in IUI conference reports. Almqvist & Wiksell, Stockholm, 1978. OCLC: 4659031.

[54] J. M. Epstein and R. Axtell. *Growing artificial societies: social science from the bottom up.* Complex adaptive systems. Brookings Institution Press, Washington, D.C, 1996.

[55] P. Erdős and A. Rényi. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959 1959.

[56] M. Eve. Deux traditions d'analyse des reseaux sociaux. *Réseaux*, 115(5):183, 2002.

[57] M. G. Everett and S. P. Borgatti. The centrality of groups and classes. *The Journal of Mathematical Sociology*, 23(3):181–201, Jan. 1999.

[58] M. G. Everett and S. P. Borgatti. Extending Centrality. In P. J. Carrington, J. Scott, and S. Wasserman, editors, *Models and Methods in Social Network Analysis*, pages 57–76. Cambridge University Press, Cambridge, 2005.

[59] J.-C. Falmagne, J.-P. Doignon, and J.-P. Doignon. *Learning spaces: interdisciplinary applied mathematics.* Springer, Heidelberg [Germany] ; New York, 2011. OCLC: ocn502033955.

[60] K. R. Finn, M. J. Silk, M. A. Porter, and N. Pinter-Wollman. The use of multilayer network analysis in animal behaviour. *arXiv:1712.01790 [nlin, physics:physics, q-bio]*, Dec. 2017. arXiv: 1712.01790.

[61] S. Fortunato. Community detection in graphs. page 103.

[62] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, Jan. 1978.

[63] C. Frelicot and F. Lebourgeois. A pretopology-based supervised pattern classifier. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, volume 1, pages 106–109, Brisbane, Qld., Australia, 1998. IEEE Comput. Soc.

[64] C. Frélicot and H. Emptoz. A pretopological approach for pattern classification with reject options. In G. Goos, J. Hartmanis, J. van Leeuwen, A. Amin, D. Dori, P. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition*, volume 1451, pages 707–715. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[65] A. Goldenberg, A. X. Zheng, and E. M. Airoldi. A Survey of Statistical Network Models. page 96.

[66] M. Granovetter. Threshold Models of Collective Behavior. *American Journal of Sociology*, 83(6):1420–1443, May 1978.

[67] A. Grow. *Agent-based modelling in population studies: concepts, methods, and applications.* Springer Berlin Heidelberg, New York, NY, 2016.

[68] G. GÜNEŞ. AGENT-BASED SIMULATION AND AN EXAMPLE IN ANYLOGIC. page 36.

[69] E. D. Habil and K. A. Elzenati. Connectedness in Isotonic Spaces. page 16.

[70] R. Hegselmann. OPINION DYNAMICS AND BOUNDED CONFIDENCE MODELS, ANALYSIS, AND SIMULATION. page 33.

[71] K. T. Ho, Q. V. Bui, and M. Bui. Dynamic Social Network Analysis Using Author-Topic Model. page 15.

[72] E. Holm, U. Lindgren, E. Lundevaller, and M. Strömgren. The SVERIGE spatial microsimulation model. In *8th Nordic Seminar on Microsimulation Models, Oslo*, pages 8–9, 2006.

[73] H.-h. Hu, J. Lin, and W.-t. Cui. Intervention Strategies and the Diffusion of Collective Behavior. *Journal of Artificial Societies and Social Simulation*, 18(3), 2015.

[74] A. Ilachinski. *Cellular automata: a discrete universe.* World Scientific, Singapore ; River Edge, NJ, 2001. OCLC: ocm47721403.

[75] G. Israel and A. Millán Gasca. *The world as a mathematical game: John von Neumann and twentieth century science.* Number v. 38 in Science networks historical studies. Birkhäuser, Basel ; Boston, 2009. OCLC: ocn277069185.

[76] M. O. Jackson and L. Yariv. Diffusion on Social Networks. page 17.

[77] D. Kempe, J. Kleinberg, and E. Tardos. Influential Nodes in a Diffusion Model for Social Networks. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming*, volume 3580, pages 1127–1138. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[78] N. N. Kenareh. Domination in Graphs. page 74.

[79] H. A. L. Kiers and I. F. of Classification Societies, editors. *Data analysis, classification, and related methods.* Studies in classification, data analysis, and knowledge organization. Springer, Berlin ; New York, 2000.

[80] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer Networks. *Journal of Complex Networks*, 2(3):203–271, Sept. 2014. arXiv: 1309.7233.

[81] J. Laborde and M. Bui. A New Pretopological Way of Identifying Spreaders in Propagation Diffusin Phenomena. In *Acta Europeana Systemica, Proceedings of the 10th congress of the European Union for Systemics*, volume 8, page 16, Brussels, Belgium, Oct. 2018.

[82] M. Lamure. *Espaces Abstraits et Reconnaissance de Formes*. PhD thesis.

[83] C. Largeron and S. Bonnevay. A pretopological approach for structural analysis. *Information Sciences*, 144(1-4):169–185, July 2002.

[84] C. Largeron and S. Bonnevay. A pretopological approach for structural analysis. *Information Sciences*, 144(1-4):169–185, July 2002.

[85] E. Lashin, A. Kozae, A. Abo Khadra, and T. Medhat. Rough set theory for topological spaces. *International Journal of Approximate Reasoning*, 40(1-2):35–43, July 2005.

[86] A. Laurent, O. Strauss, B. Bouchon-Meunier, and R. R. Yager, editors. *Information processing and management of uncertainty in knowledge-based systems: 15th international conference, IPMU 2014, Montpellier, France, July 15-19, 2014; proceedings. Pt. 2: ...* Number 443 in Communications in computer and information science. Springer, Cham, 2014. OCLC: 931888394.

[87] T. V. Le, N. Kabachi, and M. Lamure. A clustering method associated pretopological concepts and k-means algorithm. pages 529–536. WORLD SCIENTIFIC, Nov. 2007.

[88] T. V. Le, T. N. Truong, H. N. Nguyen, and T. V. Pham. An Efficient Pretopological Approach for Document Clustering. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pages 114–120, Xi'an city, Shaanxi province, China, Sept. 2013. IEEE.

[89] V. Levorato. Une méthode mixte d'analyse d'un réseau social: classification prétopologique et centralité d'intermédiarité. page 13.

[90] V. Levorato. *Contributions à la modélisation des réseaux complexes : prétopologie et applications*. PhD thesis, 2008.

[91] V. Levorato. Group Measures and Modeling for Social Networks. *Journal of Complex Systems*, 2014:1–10, 2014.

[92] V. Levorato and M. Bui. Data Structures and Algorithms for Pretopology: the JAVA based software library PretopoLib. page 14.

[93] V. Levorato and M. Bui. Modeling the Complex Dynamics of Distributed Communities of the Web with Pretopology. page 10.

[94] V. Levorato and M. Bui. Modeling the Complex Dynamics of Distributed Communities of the Web with Pretopology. page 10.

[95] Z. Lewkovicz and J.-D. Kant. A MULTIAGENT SIMULATION OF A STYLIZED FRENCH LABOR MARKET: EMERGENCES AT THE MICRO LEVEL. *Advances in Complex Systems*, 11(02):217–230, Apr. 2008.

[96] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982.

[97] D. López-Pintado and D. J. Watts. Social Influence, Binary Decisions and Collective Dynamics. *Rationality and Society*, 20(4):399–443, Nov. 2008.

[98] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[99] D. Mammass, M. E. Yassa, F. Nouboud, and A. Chalifour. A Multicriterion Pretopological Approach for Image Segmentation. page 4.

[100] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 2:415–444, 2001.

[101] S. M. Mniszewski, S. Y. D. Valle, P. D. Stroud, J. M. Riese, and S. J. Sydoriak. EpiSimS Simulation of a Multi-Component Strategy for Pandemic Influenza. page 8, 2008.

[102] M. E. J. Newman. Complex Systems: A Survey. page 10.

[103] M. E. J. Newman, A.-L. Barabási, and D. J. Watts. *The structure and dynamics of networks.* New Age International (P) Limited, Publishers, New Delhi, 2010. OCLC: 805417533.

[104] M. E. J. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3), Sept. 2003.

[105] M. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, Jan. 2005.

[106] G. H. Orcutt. A New Type of Socio-Economic System. *The Review of Economics and Statistics*, 39(2):116, May 1957.

[107] J. Oxley. What is a Matroid?

[108] J. F. Padgett. Marriage and Elite structure in Renaissance Florence, 1282-1500. page 30.

[109] J. F. Padgett and C. K. Ansell. Robust Action and the Rise of the Medici, 1400-1434. *American Journal of Sociology*, 98(6):1259–1319, May 1993.

[110] Z. Pawlak. Rough set theory and its applications. page 4.

[111] C. Petermann, S. B. Amor, and A. Bui. A pretopological multi-agent based model for an efficient and reliable Smart Grid simulation. page 7.

[112] D. R. Pritchard and E. J. Miller. Advances in population synthesis: fitting many attributes per agent and fitting to household and person margins simultaneously. *Transportation*, 39(3):685–704, May 2012.

[113] A. Riegler. Extending the Hegselmann-Krause Model II. page 14.

[114] A. Riegler and I. Douven. Extending the Hegselmann–Krause Model III: From Single Beliefs to Complex Belief States. *Episteme*, 6(02):145–163, June 2009.

[115] S. Sampson. *A novitiate in a period of change. An experimental and case study of social relationships [Ph.D. thesis], Cornell University.* PhD thesis, 1968.

[116] K. Sayadi, Q. V. Bui, and M. Bui. Distributed implementation of the latent Dirichlet allocation on Spark. In *Proceedings of the Seventh Symposium on Information and Communication Technology - SoICT '16*, pages 92–98, Ho Chi Minh City, Viet Nam, 2016. ACM Press.

[117] T. C. Schelling. Dynamic models of segregation†. *The Journal of Mathematical Sociology*, 1(2):143–186, July 1971.

[118] A. Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, Chichester ; New York, 1986.

[119] D. Shah and T. Zaman. Rumors in a Network: Who's the Culprit? *IEEE Transactions on Information Theory*, 57(8):5163–5181, Aug. 2011.

[120] M. Shokry and Y. Y. Yousif. Closure Operators on Graphs. page 9, 2011.

[121] L. Solá, M. Romance, R. Criado, J. Flores, A. García del Amo, and S. Boccaletti. Eigenvector centrality of nodes in multiplex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(3):033131, Sept. 2013.

[122] B. M. R. Stadler and P. F. Stadler. Basic Properties of Closure Spaces. page 20.

[123] Z. Suraj. An Introduction to Rough Set Theory and Its Applications. page 40.

[124] G. Toscani. Kinetic models of opinion formation. *Communications in Mathematical Sciences*, 4(3):481–496, 2006.

[125] A. Toumia and S. Szoniecky. Prétopologie et protection de la vie privée dans l'Internet des Objets. *Internet des objets*, 2(1), Feb. 2018.

[126] J. Travers and S. Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425, Dec. 1969.

[127] J. M. van Rooij and H. L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, Oct. 2011.

[128] D. Voas and P. Williamson. An evaluation of the combinatorial optimisation approach to the creation of synthetic microdata. *International Journal of Population Geography*, 6(5):349–366, Sept. 2000.

[129] E. Volz and L. A. Meyers. Susceptible-infected-recovered epidemics in dynamic contact networks. *Proceedings of the Royal Society B: Biological Sciences*, 274(1628):2925–2934, Dec. 2007.

[130] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[131] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, Cambridge, 1995. OCLC: 833770526.

[132] D. J. Watts and P. S. Dodds. Influentials, Networks, and Public Opinion Formation. *Journal of Consumer Research*, 34(4):441–458, Dec. 2007.

[133] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. 393:3, 1998.

[134] S. Wolfram. *New kind of science and a new kind of science explorer.* Wolfram Media Inc, Place of publication not identified, 2002. OCLC: 947216885.

[135] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. In R. W. Cottle, L. C. W. Dixon, B. Korte, M. J. Todd, E. L. Allgower, R. Bartels, V. Chvátal, J. E. Dennis, B. C. Eaves, R. Fletcher, M. Iri, R. G. Jeroslow, D. S. Johnson, C. Lemarechal, L. Lovasz, L. McLinden, M. W. Padberg, M. J. D. Powell, W. R. Pulleyblank, K. Ritter, R. W. H. Sargent, D. F. Shanno, L. E. Trotter, H. Tuy, R. J. B. Wets, C. Witzgall, E. M. L. Beale, G. B. Dantzig, L. V. Kantorovich, T. C. Koopmans, A. W. Tucker, P. Wolfe, and V. J. Rayward-Smith, editors, *Combinatorial Optimization II*, volume 13, pages 121–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[136] C. Wu, Y. Yue, M. Li, and O. Adjei. The rough set theory and applications. page 27.

[137] Q. Zhang, Q. Xie, and G. Wang. A survey on rough set theory and its applications. *CAAI Transactions on Intelligence Technology*, 1(4):323–333, Oct. 2016.

Résumé

En étudiant un système complexe il est naturel de s'interroger sur la structuration de ses éléments, en opposition à s'intéresser seulement aux caractéristiques individuelles des composantes. La manière traditionnelle d'aborder cette problématique a été de caractériser la structure à l'aide de un graphe, où les sommets représentent les composants du système, et un lien connecte deux composants s'il existe une relation entre l'une et l'autre. Cependant cette théorie est pertinente uniquement pour caractériser des systèmes où les rapports sous-jacents sont binaires. Nous proposons ici l'utilisation de la prétopologie afin de traiter le cas où des liens existent entre un élément et un groupe.

Nos contributions incluent la formalisation d'un espace prétopologique comme une combinaison de réseaux et de seuils, avec des règles élémentaires pour l'appartenance d'un élément à une adhérence. Ceci nous permet de stocker économiquement un espace, et d'étudier efficacement la complexité des algorithmes existants. Quelques algorithmes ont été améliorés.

Quelques applications on été développées. La première a été la création d'une librairie prétopologique en Python. La librairie est ensuite utilisée pour étudier quelques problèmes liés à la diffusion au sein d'un système. La troisième et dernière application consiste à utiliser la prétopologie dans le contexte du partitionnement de données.

Mots Clés

Prétopologie, Algorithmique, Réseaux complexes, Diffusion

Abstract

When we study a complex system it is natural to be interested in the structure of its elements, as opposed to just caring about the individual characteristics of the components. The traditional way to deal with this issue has been through a characterization of the structure by the use of graphs, where nodes represent the components of the system, and an edge exists between two of them if there is a relation connecting them. The theory is nevertheless only appropriate for the description of systems with binary underlying relations between the components. We propose here the use of pretopology to effectively treat the case where connections exist between an element and a group.

Our contributions include a formalization of a pretopological space in terms of a simple group of rules over a set of networks. This allow us to economically store a pretopological space, and to effectively study the complexity of the algorithms that have been proposed. Some of these algorithms were improved.

Some applications are developed. The first of them is a Python library where all of the previously reviewed algorithms are implemented. After presenting in a general manner the faculties of the library, we use it to study some diffusion phenomena with the help of some agent based models. We finish using some of the notions introduced here to develop a clusterization algorithm, that not only performs on a par with the state of the art on some artificial geometrical data, but also has the advantage of being immediately generalizable to non-metrical spaces.

Keywords

Pretopology, Algorithmic, Complex Networks, Diffusion