Università degli Studi di Salerno

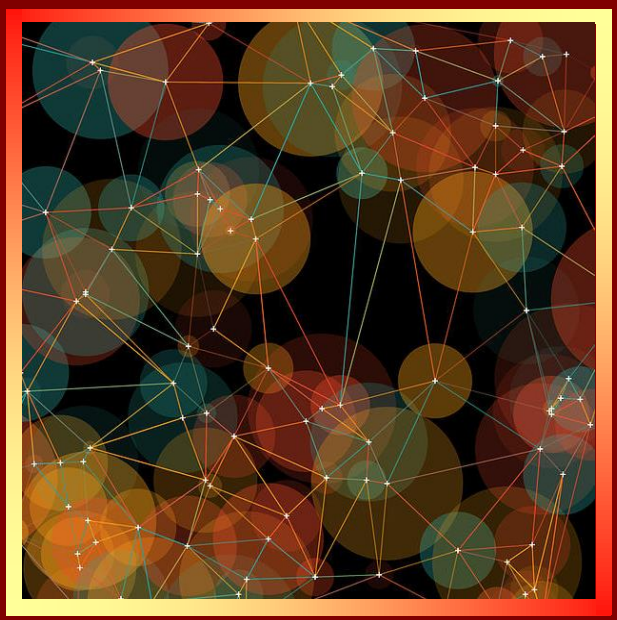
Dipartimento di Informatica

Dottorato di Ricerca in Informatica – XXXIV Ciclo

Tesi di Dottorato/Ph.D. Thesis

Beyond Pairwise Relationships: Modeling Real-world Dynamics Via High-order Networks

Alessia Antelmi



Supervisor: **Prof. Vittorio Scarano**

Ph.D. Program Director: **Prof. Andrea De Lucia**

AA 2020/2021

Curriculum Computer Science and Information Technology

Università degli Studi di Salerno

Dipartimento di Informatica

Dottorato di Ricerca in Informatica
XXXIV Ciclo

Tesi di Dottorato / Ph.D. Thesis

Beyond Pairwise Relationships: Modeling Real-world Dynamics via High-order Networks

Alessia Antelmi

Supervisor: **Prof. Vittorio Scarano**

Ph.D. Program Director: **Prof. Andrea De Lucia**

A.A. 2020/2021

To you, who believe in me.

Acknowledgments

For sure, the following words cannot truly express the immense gratitude I have for all the fantastic people who shared with me this little piece of my life.

Foremost, I wish to thank professor Vittorio Scarano, not only my supervisor but my guide ever since before the beginning of this journey. I owe him all the experiences I had the opportunity to live, and I know he was always there for me. I am also profoundly thankful to professor Gennaro Cordasco, irreplaceable mentor and wise guide. Last but not least, my greatest gratitude goes to dr. Carmine Spagnuolo, big brother in this experience, for his invaluable guidance and precious friendship. Thanks to *ISISLab*, always my second home.

A special thank goes to my true friends, too often too far away to hug, but always there to have my back.

Mirko, another significant thank is for you, for your precious advice, and for simply being my rock in the storm.

The final thank goes to my family and my grandparents. Here, every word is superfluous - my greatest desire is to make you proud of me, now and in the future.

Abstract

Every single person, animal, or thing we can see in the world around us is part of a broader collection of components that can spontaneously self-organize to exhibit non-trivial global structures and behaviors at larger scales, often without external intervention, central authorities, or leaders. The properties of the collection these components give life cannot be understood or predicted from the full knowledge of its elements alone. Each collection is an example of complex systems whose behavior is intrinsically challenging to model due to the high non-linearity of the interactions between its constituents.

Traditionally, complex systems have been successfully studied through graphs abstracting the underlying network with vertices and edges connecting pairs of interacting components. Over the years, the scientific community has enriched the graph modeling framework for better capturing the richness of the interactions among such units. However, graphs have a substantial limitation encoded in their nature: they exclusively capture pairwise interactions. Yet, many complex systems are characterized by group interactions that cannot be described simply in terms of dyads. Studying such systems hence require new mathematical frameworks and scientific methodologies for its investigation.

Hypergraphs are the perfect candidates to tackle this task. A hypergraph is a generalization of a graph, where a (hyper)edge allows the connection of an arbitrary number of vertices. However, the powerful expressiveness of hypergraphs has a few drawbacks: dealing with the complexity of such data structures and the lack of appropriate tools and algorithms for their study. For this reason, hypergraphs have been little used in literature in favor of their graph-counterpart. Recently, this trend

has been drifting, thanks to an increasing number of systematic studies demonstrating that considering the higher-order structure of complex systems can enhance our modeling capacities and help us understand and predict their dynamical behavior.

This dissertation fits in this broad context of modeling complex systems with the general objective of formalizing and implementing more expressive network models. Specifically, the whole work is rooted in understanding how much and when we need high-order information conveyed by hypergraphs. The contribution described in this thesis can be grouped according to three macro topics.

I. Tools for hypergraphs. Motivated by the lack of a comprehensive and efficient hypergraph-specific library and the need for software libraries designed to perform operations directly on hypergraphs, we developed SimpleHypergraphs.jl, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing. This dissertation describes the main motivations behind creating SimpleHypergraphs.jl, the library's design choices, and its memory model. We further illustrate the functionalities offered by the software, including graph transformations and hypergraph visualization methods. We also present two case studies with the twofold objective of demonstrating how it is possible to exploit the proposed library and comparing hypergraphs with their corresponding graph counterpart to explore whether high-order structures convey more information in addressing specific tasks. Contextually, we also describe a generalized version of the label propagation algorithm for community detection suitable for hypergraphs.

The second and third topics addressed instantiate the initial broad research question into two principal research directions, both tied to the concept of diffusion. Informally, the term *diffusion* means the process according to whom an entity (e.g., information) spreads within a network,

moving from node to node or group of nodes, through their interactions. Under this umbrella, this thesis focuses on studying *social influence propagation* and *epidemic spreading processes* within a population.

II. Social influence on high-order networks. In the context of social influence diffusion, we propose the formal definition of a high-order diffusion process, the generalization of a well-known graph problem to hypergraphs, and a set of heuristics to tackle it. Specifically, in this thesis, we first introduce the motivation behind this line of study and discuss a new high-order diffusion model with linear thresholds that mimics real-world social dynamics, where individuals influence the group they belong to, but - in turn - the group itself influences their choices. We further introduce the formal definition of the Target Set Selection problem on hypergraphs (TSSH), a key algorithmic question in information diffusion research, whose goal is to find the smaller set of vertices that can influence the whole network according to the diffusion model defined. Since the TSSH problem is NP-hard, we propose four heuristics to address it and extensively evaluate these algorithms on random and real-world networks.

III. Epidemic dynamics on temporal high-order networks. From the perspective of epidemic dynamics, we propose the formal definition of time-varying hypergraphs (TVHs), the introduction of direct and indirect interactions when studying an epidemic spreading via a TVH contact network, and an epidemic diffusion algorithm built on top of TVHs and direct and indirect contagion pathways. This dissertation motivates why one should use (temporal) hypergraphs rather than (temporal) graphs to analyze epidemic spreading processes. We then introduce the formal definition of temporal hypergraphs, describe a high-order SIS compartmental equation model suitable for TVHs, and discuss how we assembled these elements into an agent-based framework. We further present a sensitivity analysis of the TVH model to the epidemic parameters and different discretization of the time intervals when direct or indirect contacts may happen.

Built on top of the TVH model, we also propose a fine-grain modeling methodology for Non-Pharmaceutical Interventions (NPIs). We first motivate why one should evaluate such epidemic control strategies in the framework of agent-based models (ABMs) and high-order interactions. We then delve into reviewing personal protective, environmental, and social distancing measures and how they can be embedded into an epidemiological model based on high-order networks, ABMs, and the SIS equation-based model. We further describe how we formally enriched the TVH modeling framework to support the evaluation of NPIs. After assessing the ability of each intervention in controlling an epidemic propagation, we discuss a multi-objective optimization framework, which, based on the epidemiological data, calculates the NPI combination that should be implemented to minimize the spread of an epidemic as well as the damage due to the intervention.

Contents

PART

I

Introduction & Background

Summary

1

Introduction

> Do you wish me a good morning,
> or mean that it is a good morning whether I want it or not;
> or that you feel good this morning;
> or that it is a morning to be good on?
>
> *The Hobbit*
> J. R. R. Tolkien

In short

A flock, a social or economic organization, or still the Earth's global climate: what do all these elements have in common? They all are a conglomerate of interconnected and interdependent parts. Their interactions give life to behaviors and effects incomprehensible if we look at a single component at a time. No matter how good or accurate our knowledge at the individual unit level is: we cannot explain viral rumors spreading across societies from individual human psychology or predicting next week's forecast considering only the atmospheric pressure. Those elements are examples of complex systems whose behavior is intrinsically challenging to model due to the high non-linearity of the interactions between its components. After scientists abandoned the reductionism approach, such complexity has been recognized as one of the principles governing real-world dynamics [10].

Graphs have emerged as an ideal modeling tool to abstract the elements of a complex system and its interactions, with applications spanning the full spectrum of science, from fundamental physics all the way to the social sciences [28]. The application of such instruments to model and analyze real-world systems has to account for various existing relations [49]. Simply representing the elementary unit of a system as a collection of entities (vertices) and describing the interactions between pairs of such units (edges) may result in an abstraction not expressive enough. During the past years, the effort of the research community has been devoted to formalizing and developing mathematical tools to include the concepts of direct interaction (directed graphs), importance and strength of a relation (weighted [29, 31] and signed graphs [175]), nature and dynamicity of the interactions themselves (multi-layer [119, 40] and temporal graphs [94]). Still, an underlying question remains.

> *Are these models powerful enough to provide a complete description of a complex system?*

This dissertation fits in this broad context of modeling complex systems with the general objective of formalizing and implementing even more expressive network models. This chapter opens this thesis by introducing the motivations in Section 1.1 and outlining the main contributions in Section 1.2. It finally presents how this dissertation is organized in Section 1.3.

1.1 Motivation

Probably, the answer to the previous question is no. These models still have a substantial limitation encoded in their nature: they exclusively capture pairwise interactions. Yet, many complex systems are characterized by group interactions. Just think - for instance - to biological processes involving more than two participating partners, like a metabolic reaction such as $A + B \rightarrow C + D$ (involving four species), or a protein complex consisting of more than two proteins. We may also think about a trivial

example of a co-authorship network, where researchers publish articles in groups of two or more people. How can we go back to which author subgroup has worked on the same paper? Obviously, this information can be stored externally or within the network itself, but querying such structures may not be trivial or efficient. Similar situations, characterized by group (or higher-order or many-to-many) interactions, exist not only in biology or in social systems but also in ecology and neuroscience [32].

Can we instead find mathematical frameworks that can explicitly and naturally describe many-to-many interactions?

Hypergraphs are the perfect candidates to answer such a question. A hypergraph is a generalization of a graph, where a (hyper)edge allows the connection of an arbitrary number of vertices (see Figure 1.1a). However, the powerful expressiveness of hypergraphs has a few drawbacks: dealing with the complexity of such data structures and the lack of appropriate tools and algorithms for their study. For this reason, hypergraphs have been little used in literature in favor of their graph-counterpart. There exist several graph-based representations of a hypergraph on which a plethora of classical network science techniques are usually applied. Nevertheless, these models inevitably lose information when encoding the high-order nature of the underlying data.

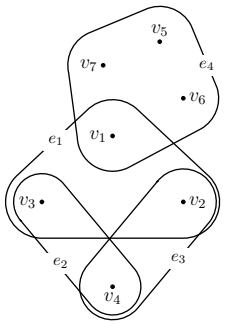
Line graphs - in which each vertex represents a hyperedge and two vertices are linked if the corresponding hyperedges share at least a vertex - are a typical example of hypergraph to graph transformation (see Figure 1.1b). Still, line graphs have two significant limitations [4]. First, distinct hypergraphs can have identical line graphs as they lose the information about the composition of each hyperedge, storing only whether two hyperedges intersect but not in which manner. Second, sparse hypergraphs can yield relatively dense line graphs as a vertex of degree d in the hypergraph yield $\binom{d}{2}$ edges in its line graph, thus making line graphs challenging to analyze or store in computer memory.

Clique graphs embody another example of this transformation (see Figure 1.1c). This representation directly encodes the idea that each mem-

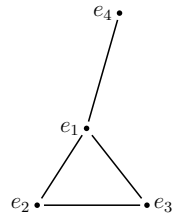
ber of a relationship interacts with every other. Formally, this concept is translated by considering each hyperedge a complete sub-graph. Clique graphs share the same limitations as line graphs. The major drawback of using such a transformation is that clique graphs completely lose the notion of groups since pairwise connections substitute each high-order interaction. Consequently, we have a high probability of materializing interactions that did not exist in the original hypergraph. This intuitive concept of losing the notion of groups is formalized by the fact that different hypergraphs can be transformed in the same clique graph; hence, we cannot uniquely reconstruct the original hypergraph from its clique graph. Further, like line graphs, clique graphs can yield computational issues as each hyperedge of size k is transformed into $\frac{k\times(k-1)}{2}$ edges.

Conversely, bipartite graph representations effectively describe group interactions (see Figure 1.1d). In this model, one vertex set corresponds to the hypergraph's vertices, the other to the hyperedges. Hence, a link in this graph connects a vertex to the interactions - of arbitrary order - in which it takes part. This model is widely used as a network science tool thanks to its capability to mimic most interaction structures. However, also bipartite graphs have a critical shortcoming inherent in their structure [32]. Vertices in the original system do not interact directly anymore as the interaction layer always mediates their relationship. This interaction structure implies that any measure or dynamic process defined on the bipartite representation must consider this additional complexity.

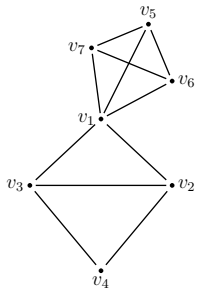
In a nutshell, despite graphs' powerful capability in capturing many properties of complex interacting systems, they are intrinsically limited in explicitly describing group interactions, and there is no possibility to reconstruct high-order relationships from their structure. Even when networks can provide information on many-to-many interactions, as in the case of bipartite graphs, this expressiveness is rewarded with additional complexity. Recently, the trend of using hypergraphs to graph representations is drifting, thanks to an increasing number of systematic studies demonstrating how this transformation process may imply an inevitable loss of information, leading to wrong interpretations afterward [195, 32].



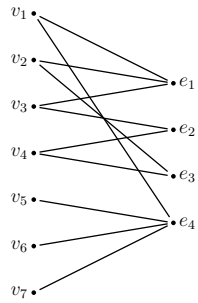
(a) Hypergraph.



(b) Line graph.



(c) Clique graph.



(d) Bipartite graph.

Figure 1.1: Hypergraph to graph transformations.

All the above considerations stand as a general motivation underlying this dissertation, answering why hypergraphs should be used as data structures to abstract and analyze real-world high-order interactions. The need to use hypergraphs (not through transformation to graphs) directly motivates the development of specific tools to handle such structures, and their recent growth in the research landscape emphasizes the existing gap to state-of-the-art instruments for dealing with graphs. Another direct implication is the need to re-think and re-discuss phenomena traditionally studied via the lens of graphs. Among those, we can surely mention dynamical processes emulating human behaviors, which are the focus of many studies where social relationships and interactions are typ-

ically considered an underlying structure [30]. This kind of interactions lends itself well for testing higher-order approaches since individuals can interact in pairs or groups. Towards this, dynamic processes over hypergraphs could account for the higher-order effects that the non-pairwise interactions might lead to [32].

1.2 Contributions

This dissertation is rooted in understanding how much and when we need high-order information conveyed by hypergraphs. Specifically, this broad research question is instantiated into two principal research directions, both tied to the concept of diffusion. Informally, the term *diffusion* means the process according to whom an entity (e.g., information) spreads within a network, moving from node to node or group of nodes, through their interactions. Under this umbrella, this thesis focuses on studying *social influence propagation* and *epidemic spreading processes* within a population. Dealing with such problems requires having the appropriate tools to easily implement and efficiently run the experiments to validate models and hypotheses. Towards this direction, this thesis also introduces a new hypergraph software framework for the Julia language.

The contributions of this dissertation can be grouped according to the three macro topics addressed and are summarized in the following.

Tools for hypergraphs.

- The design and implementation of **SimpleHypergraphs.jl**, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing;
- The proposal of a **generalized version of the label propagation algorithm** for community detection suitable for hypergraphs;
- The definition of the notion of **s-betweenness centrality** for hypergraphs;

- The **comparison** of hypergraphs with their corresponding graph counterpart to explore whether high-order structures convey more information in addressing specific tasks.

Social influence diffusion on high-order networks.

- The formal definition of a **high-order diffusion process** for hypergraphs based on the linear threshold model, mimicking real-world social dynamics, in which individuals influence the group they belong to, but - in turn - the group itself influences their choices;
- The formal definition of the **Target Set Selection problem on hypergraphs** (TSSH), whose goal is to find the smaller set of vertices that can influence the whole network;
- The proposal of **four greedy-based heuristics** to address the TSSH problem, followed by an optional optimization procedure;
- Extensive **evaluation** of the heuristics on random and real-world networks.

Epidemic dynamics on high-order temporal networks.

- The formal definition of **time-varying hypergraphs** (TVHs);
- The introduction of **direct and indirect interactions** when studying an epidemic spreading via a TVH contact network;
- The proposal of a **high-order epidemic diffusion algorithm** built on top of TVHs and direct and indirect contagion pathways;
- **Sensitivity analysis** of the TVH model to the epidemic parameters and different discretization of the time intervals when direct or indirect contacts may happen;
- The formal definition of **Non-Pharmaceutical Interventions** (NPIs), described by the WHO in [148], for the epidemiological framework based on TVHs;

- **Evaluation of each NPI** applying the SIS compartmental equation-model into an agent-based model that exploits the high-order epidemic diffusion algorithm to simulate interactions between agents and locations;
- The **design and implementation of a genetic algorithm-based methodology** to optimize which NPI combination has to be adopted when contrasting objectives are considered.

Table 1.1 summarizes the contributions discussed in this dissertation, including the peer-reviewed venues where each piece of work has been published. All code produced for models and experiments is freely available on the GitHub platform.

1.3 Document Structure

This thesis is divided in five main parts:

PART I, beginning with this chapter, provides the reader with all the background knowledge necessary to fully understand the remainder of the thesis and contextualizes the contributions discussed in the following parts. Specifically, Chapter 2 introduces the concepts and the notation we will use throughout this dissertation. Chapter 3, Chapter 4, and Chapter 5 thoroughly describe the current state-of-the-art concerning the proposed contributions. Finally, Chapter 6 describes the data sets used in the experimental part of this thesis.

PART II introduces the first contribution of this dissertation: SimpleHypergraphs.jl, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing. Chapter 7 describes the main motivations behind creating SimpleHypergraphs.jl, the library's design choices, and its memory model. It further illustrates the functionalities offered by the software, including transformation to graphs and hypergraph visualization methods. Finally, it presents two case studies to demonstrate

Table 1.1: Summary of the contributions presented in this dissertation.

Topic	Pub. Type (Year)	Title	Contribution
Tools for hypergraphs	Workshop WAW (2019)	SimpleHypergraphs.jl— Novel Software Framework for Modelling and Analysis of Hypergraphs [13]	First release of `SimpleHypergraphs.jl` with a set of basic functionalities and a practical use case.
	Journal Internet Mathematics (2020)	Analyzing, Exploring, and Visualizing Complex Networks via Hypergraphs using SimpleHypergraphs.jl [14]	New version of the library, with hypergraph specific algorithms and visualization functionalities, along with two additional use cases.
Social Influence	Workshop WAW (2020)	Information Diffusion in Complex Networks: A Model Based on Hypergraphs and Its Analysis [17]	Formal definition of a high-order diffusion process and the TSS problem on hypergraphs, with three greedy-based additive heuristics to address it.
	Journal Entropy (2021)	Social Influence Maximization in Hypergraphs [18]	Enhancement of the three heuristics and proposal of a new subtractive heuristic. Comprehensive experiments on real-world hyper-networks.
Epidemic Dynamics	Conference AAMAS (2020)	A Design-Methodology for Epidemic Dynamics via Time-Varying Hypergraphs [16]	Modeling high-order epidemic processes via TVHs: formal definition and evaluation.
	Journal IEEE Access (2021)	Modeling and Evaluating Epidemic Control Strategies with High-Order Temporal Networks [15]	Introduction of NPIs within the TVH framework: formal definition and evaluation.

how to exploit the proposed library and compare hypergraphs with their corresponding graph counterpart to explore whether high-order structures convey more information in addressing specific tasks.

PART III presents the second contribution of this dissertation. Chapter 8 describes a new linear threshold high-order diffusion model and the formal definition of the target set selection problem on hypergraphs. Then, it introduces four heuristics to address it and finally provides an extensive evaluation of the proposed algorithms on random and real-world hyper-networks.

PART IV discusses the third contribution of this dissertation. Chapter 9

presents the formal definition of time-varying hypergraphs (TVHs), the introduction of direct and indirect interactions when studying an epidemic spreading via a TVH contact network, and an epidemic diffusion algorithm built on top of TVHs and direct and indirect contagion pathways. Chapter 10 discusses a fine-grain modeling methodology for NPIs built on top of the TVH model. It describes how the TVH modeling framework has been formally enriched to support the evaluation of NPIs and finally discusses a multi-objective optimization framework, which, based on the epidemiological data, calculates the NPI combination that should be implemented to minimize the spread of an epidemic as well as the damage due to the intervention.

PART V concludes this dissertation. Chapter 11 wraps up the contributions, the achieved research outcomes, and delineates the potential impact of this research. It finally discusses possible future directions on each specific topic addressed.

1.3.1 How to Read this Dissertation

In this section, we give some insights about the structure of the document to guide the reader through this dissertation.

General structure.

Part I only discusses the mathematical background and frames each specific topic in the current literature. Part II, III, and IV discuss the contributions presented in this thesis.

Chapters' structure.

The chapters detailing the dissertation's contributions follow the same structure by describing:

- The motivation behind the specific problem addressed;
- The contribution;

- A state-of-the-art comparison;
- The experiment setting;
- Results;
- Remarks summarizing the chapter content.

Special content.

Some key elements presented in this thesis are emphasized in special boxes.

Definition 1.1 : A first definition

Definitions are reported in yellow boxes.

A sample assumption

Assumptions are described in light blue boxes.

A special information

Particular information are described in green boxes.

A sample observation

Observations are reported in purple boxes.

A sample exception

Exceptions are stressed in orange boxes.

Some limitations

Limitations are highlighted in magenta boxes.

Code and data information

Code and data links are detailed in blue boxes.

2

Definitions and Notation

The origins of graph theory are humble, even frivolous.

Graph Theory 1736-1936
N. L. Biggs, E. K. Lloyd, and R. J. Wilson

In short

This chapter introduces the concepts and the notation we will use throughout this dissertation. Starting from formally defining hypergraphs in Section 2.1, we will then describe how these structures can be represented via pairwise relationships in Section 2.2. Finally, we list the notation used in Section 2.3.

2.1 Hypergraphs: Basic Concepts

Hypergraphs are the natural representation of a broad range of systems where group (or high-order or many-to-many) relationships exist among their interacting parts. Technically speaking, a hypergraph is a generalization of a graph, where a (hyper)edge allows the connection of an arbitrary number of nodes. As a direct consequence, many of the definitions of graphs carry verbatim to hypergraphs [47].

Definition 2.1 : Hypergraphs

A *hypergraph* H, denoted with $H = (\mathcal{V}, E = (e_i)_{i \in \mathcal{I}})$, on a finite set $\mathcal{V}$ and a finite set of indexes $\mathcal{I}$ is a family $(e_i)_{i \in \mathcal{I}}$ of subsets of $\mathcal{V}$ called hyperedges.

The *order* of a hypergraph $H = (\mathcal{V}, E)$ is the cardinality of $\mathcal{V}$, i.e., $|\mathcal{V}| = n$; while the *size* of H is the cardinality of E, i.e. $|E| = m$. By definition,

- An *empty hypergraph* is a hypergraph such that $\mathcal{V} = \emptyset$ and $E = \emptyset$;
- A *trivial hypergraph* is a hypergraph such that $\mathcal{V} \neq \emptyset$ and $E = \emptyset$.

Assumptions on the structure of the hypergraph H

In the remainder of this thesis, hypergraphs have a non-empty set of vertices, and a non-empty set of hyperedges.

A vertex v is isolated if $v \in \mathcal{V} \setminus \bigcup_{i \in \mathcal{I}} e_i$. If $\bigcup_{i \in \mathcal{I}} e_i = \mathcal{V}$, the hypergraph has no *isolated vertices*. Two vertices in a hypergraph are *adjacent* if there is a hyperedge containing both vertices. In particular, if $\{v\}$ is a hyperedge, then v is adjacent to itself.

A hyperedge $e \in E$ such that $|e| = 1$ is a *loop*. If the family of hyperedges is a set, i.e., if $i \neq j \iff e_i \neq e_j$, then H has *no repeated hyperedges*. A hyperedge $e \in E$ is *incident* with a vertex $c \in \mathcal{V}$ if $v \in e$. Two hyperedges in a hypergraph are *adjacent* if their intersection is not empty, i.e., if they are incident with at least a common vertex.

In the literature, it is possible to find slightly different definitions of a hypergraph. For instance, Berge [36] prohibits empty hyperedges and

isolated vertices. On the contrary, Katona [116] allows empty hyperedges and isolated vertices, but defines $E = \{e_1, e_2, \ldots, e_m\}$ as a set and explicitly prohibits pairs of duplicated hyperedges $e_i = e_j$ for $i \neq j$. Bretto [47] and Aksoy et al. [4] provide the most general definition for such structures, accounting for isolated vertices, as well as empty, duplicated, and singleton hyperedges. This choice does not impose any constraint on the nature of the data to model, facilitating the fitting of hypergraphs to real-world networks. In this thesis, we will follow Bretto's definitions [47].

The star $H(v)$ centered in v is the family of hyperedges containing v. If the hypergraph has no repeated hyperedges, the degree of v is denoted by $deg(v) = |H(v)|$. If each vertex has the same degree d, we say that the hypergraph is *d-regular*, i.e., for every $v \in \mathcal{V}$, $deg(v) = d$. We denote with $D(H)$ the maximal degree of a hypergraph H.

The *rank* $r(H)$ of H is the maximum cardinality of a hyperedge in the hypergraph, $r(H) = \max_{i \in \mathcal{I}} |e_i|$. The minimum cardinality of a hyperedge is the *co-rank* $cr(H)$ of H, $cr(H) = \min_{i \in \mathcal{I}} |e_i|$. If $r(H) = cr(H) = k$, then the hypergraph is *k-uniform*. In other words, when all hyperedges have the same degree k, i.e., $|e| = k \; \forall e \in E$, we say that H is a k-uniform hypergraph.

A *simple* hypergraph is a hypergraph $H = (\mathcal{V}, E)$ such that $e_i \subseteq e_j \implies i = j$ (no hyperedge is properly contained in any other). A simple hypergraph has no repeated hyperedges. A hypergraph is *linear* if it is simple and $|e_i \cap e_j| \leq 1$ for all $i, j \in \mathcal{I}$ where $i \neq j$.

A Sample Hypergraph

Let C be a cinema with $m \geq 1$ movies, $M_1, M_2, \ldots, M_m$, and let $\mathcal{V}_c$ be the set of people going at the cinema. Assume that each movie is seen by one person at least. We can build the hypergraph $C = (\mathcal{V}_c, E_c)$ as follows:

- The set of vertices $\mathcal{V}_c$ is the set of people who are in the cinema;

- Each hyperedge e_i, $i \in \{1, 2, \ldots, m\}$, is the subset of people watching the movie M_i.

Figure 2.1 visualizes an instance of the above hypergraph, considering four movies ($|E_c| = m = 4$) and seven people ($|\mathcal{V}_c| = n = 7$). Specifically, the hypergraph $C = (\mathcal{V}_c, E_c)$ can be defined as follows:

- $\mathcal{V}_c = \{p_1, p_2, \ldots, p_7\}$;
- $E_c = \{M_1, M_2, M_3, M_4\}$, where $M_1 = \{p_1, p_2, p_3\}$, $M_2 = \{p_2, p_3\}$, $M_3 = \{p_3, p_5, p_6\}$, and $M_4 = \{p_4\}$.

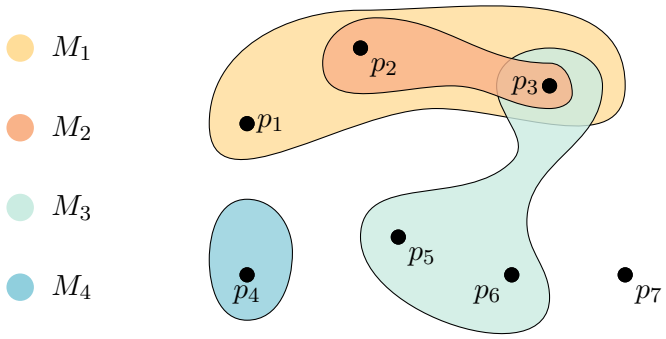


Figure 2.1: The example hypergraph C has seven vertices, four hyperedges, one loop (p_4), and one isolated vertex (p_7). The rank $r(C) = 3$, the co-rank $cr(C) = 1$. The degree of p_3 is 3.

2.1.1 Matrix Notation

Let $H = (\mathcal{V}, E)$ be a hypergraph, with $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and $E = (e_1, e_2, \ldots, e_m)$. The structure of H may also be represented by an *incidence matrix* $\mathbf{H} \in \{0, 1\}^{|\mathcal{V}| \times |E|}$, with each entry $\mathbf{H}(v_i, e_j)$ indicating whether the vertex v_i is in the hyperedge e_j, with $1 \leq i \leq n$ and $1 \leq j \leq m$. Formally,

$$\mathbf{H}(v_i, e_j) = \begin{cases} 1, & \text{if } v_i \in e_j \\ 0, & \text{otherwise} \end{cases}.$$

The degree $deg(v)$ of a vertex v and the degree $\kappa(e)$ of a hyperedge e can be rewritten as

$$\deg(v) = \sum_{e \in E} \mathbf{H}(v, e) \text{ and } \kappa(e) = |e| = \sum_{v \in \mathcal{V}} \mathbf{H}(v, e),$$

respectively.

For example, the incidence matrix of the hypergraph C in Figure 2.1 is the 7×4 matrix:

$$\mathbf{C} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{array}{cccc} M_1 & M_2 & M_3 & M_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

As for graphs, the *adjacency matrix* $\mathbf{A}(H)$ of H is a square matrix whose rows and columns are indexed with the vertices of H and for all $u, v \in \mathcal{V}, u \neq v$ the entry $\mathbf{A}(u, v) = |e \in E : u, v \in e|$ and $\mathbf{A}(u, u) = 0$. The matrix $\mathbf{A}$ is symmetric and all $\mathbf{A}(u, v)$ belong to $\mathbb{N}$. It is worth noting that $\mathbf{A}$ corresponds the matrix of a multi-graph. The adjacency matrix of the hypergraph C in Figure 2.1 is the 7×7 matrix:

$$\mathbf{A}(C) = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

2.1.2 The Dual of a Hypergraph

Loosely speaking, the dual of a hypergraph is the hypergraph constructed by swapping the roles of vertices and hyperedges. Formally, let $H = (\mathcal{V}, E)$ be a hypergraph with vertex set $\mathcal{V} = \{v_i\}_{i\in[1,n]}$ and family of hyperedges $E = (e_j)_{j\in[1,m]}$. The *dual* hypergraph $H^* = (\mathcal{V}^*, E^*)$ of H is the hypergraph such that:

- The set of vertices, $\mathcal{V}^* = \{v_1^*, v_2^*, \ldots, v_m^*\}$ is in bijection $g : E \to \mathcal{V}^*$ with the set of hyperedges E;
- The set of hyperedges is given by $e_1^* = V_1, e_2^* = V_2, \ldots, e_n^* = V_n$, where $e_i^* = V_i = \{g(e_j) = v_j^* : v_i \in e_j\}$. In words, each vertex v_i of H corresponds to an hyperedge e_i* of H^* , where e_i* is the set of hyperedges of H (i.e., vertices of the dual) that contain v_i.

Without loss of generality, and when there is no ambiguity, we identify $\mathcal{V}^*$ with E, thus having $e_i^* = V_i = e_j : v_i \in e_j$, for $i \in \{1, 2, \ldots, n\}$. In other words, there is a bijection $f : \mathcal{V} \to E^*$ from the vertices $\mathcal{V}$ of H to the hyperedges E^* of H^*.

The incidence matrix of H^* is the transpose of the incidence matrix $\mathbf{A}^\mathsf{T}$ of the hypergraph H. Thus, we have for $v_j^* \in \mathcal{V}^*$ and $e_i^* \in E^*$, $v_j^* \in e_i^* \iff \mathbf{A}(i, j) = 1$. As a consequence, $(H^*)^* = H$. For instance, the incidence matrix of the dual hypergraph C^* (see Figure 2.2) of the hypergraph C (see Figure 2.1) is the 4×7 matrix:

$$\mathbf{C}^* = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

We can further observe that two vertices belonging to the same set of edges in H correspond to multi-edges in H^* and isolated vertices in H correspond to empty edges in H^* . The generality of Definition 2.1 in permit-

ting multi-edges, empty edges, and isolated vertices ensures the dual of a hypergraph is also a hypergraph.

At this point, we can re-write the definition of the dual of H in a more intuitive way as

$$H^* = (\mathcal{V}^* = E, E^* = (H(v))_{v \in \mathcal{V}}).$$

Clearly, the equivalence $D(H) = r(H^*)$ holds.

Example

Recalling the hypergraph C described in the paragraph §A Sample Hypergraph and shown in Figure 2.1, let us consider the dual hypergraph C^* of C. The hypergraph C^* is composed as follows:

- $\mathcal{V}_c^* = \{M_1, M_2, M_3, M_4\}$;
- $E_c^* = (p_1, p_2, \ldots, p_7)$, where $p_1 = \{M_1\}$, $p_2 = \{M_1, M_2\}$, $p_3 = \{M_1, M_2, M_3\}$, $p_4 = \{M_4\}$, $p_5 = \{M_3\}$, $p_6 = \{M_3\}$, and $p_7 = \emptyset$.

Figure 2.2 shows the hypergraph C^*. Table 2.1 lists some relations between the hypergraph H and its dual H^* .

Table 2.1: Relations existing between a hypergraph H and the corresponding dual hypergraph H^* .

H	H*
$v \in e$	$g(e) \in f(v)$
$deg(v)$	$\lvert f(v) \rvert$
$\lvert e \rvert$ or $\kappa(e)$	$deg(g(e))$
k-uniform	k-regular
d-regular	d-uniform

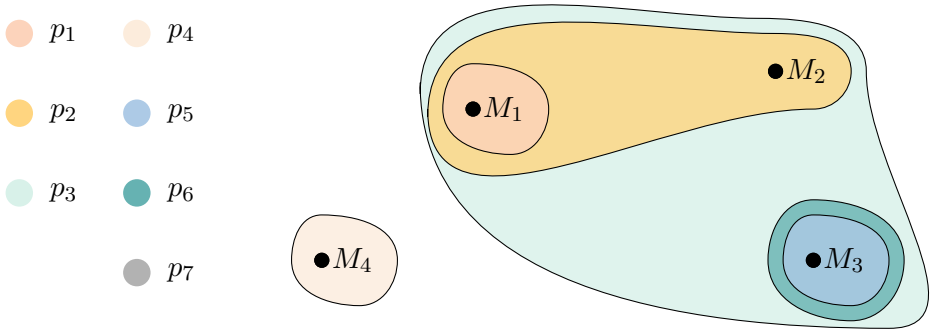


Figure 2.2: The dual hypergraph C^* of the hypergraph C in Figure 2.1. The hypergraph C^* has four vertices, seven hyperedges, four loops (M_1, M_3, M_4), no isolated vertices, and an empty hyperedge (p_7). The rank $r(C) = 3$, the co-rank $cr(C) = 1$. The degree of M_3 is 3.

2.1.3 Weighted Hypergraphs

As happens for graphs, hypergraphs may also have a weight associated with their elements describing either the importance of a relation or the relevance of a specific element within a group.

In a *weighted* hypergraph, denoted by a tuple $H = (\mathcal{V}, E, \mathbf{W})$, each hyperedge $e \in E$ has a weight $w(e)$, representing the importance of that relation in the whole hypergraph. $\mathbf{W} \in \mathbb{R}^{|E|\times|E|}$ denotes the diagonal matrix (with 0s in all off-diagonal entries) of the hyperedge weights, i.e.,

$$\text{diag}(\mathbf{W}) = [w(e_1), w(e_2), ..., w(e_{|E|})].$$

Moreover, in a hypergraph with *edge-dependent vertex weights* [55], denoted with $H = (\mathcal{V}, E, \mathbf{W}, \lambda)$, each entry of the incidence matrix $\mathbf{H}(v, e)$ represents the weight $\lambda_e(v)$ of the vertex v in the hyperedge e. This value can be interpreted as the contribution of the element v to the relation e.

In a weighted hypergraph, the degree $deg(v)$ of a vertex v and the de-

gree $\kappa(e)$ of a hyperedge e can be rewritten as

$$\deg(v) = \sum_{e \in E} w(e)\mathbf{H}(v, e) \quad \text{and} \quad \kappa(e) = \sum_{v \in \mathcal{V}} \lambda_e(v)\mathbf{H}(v, e),$$

respectively.

Assumptions on the nature of the hypergraph H

As for classical graphs, we can also find in the literature either directed and heterogeneous hypergraphs. We will not delve into introducing such structures as the work described in this dissertation relates only to *undirected* and *homogeneous* hypergraphs.

2.1.4 Paths and Connected Components

Let $H = (\mathcal{V}, E)$ be a hypergraph without isolated vertices. A *path* P in H from u to v, is a vertex-hyperedge alternating sequence:

$$u = u_1, e_1, u_2, e_2, \ldots, u_s, e_s, u_{s+1} = v,$$

such that

- $u_1, u_2, \ldots, u_s, u_{s+1}$ are distinct vertices with the possibility that $u_1 = u_{s+1}$;
- $e_1, e_2, \ldots, e_s$ are distinct hyperedges;
- $u_i, u_{i+1} \in e_i, (i = 1, 2, \ldots, s)$.

The integer s is the *length* of the path P. If $u_1 = u_{s+1} = v$, then P is a *cycle*.

As for graphs, we say that P connects two vertices u and v if there is a path from u to v and vice-versa. A hypergraph is *connected* if for any pair

of vertices, there is a path connecting them; it is *disconnected* otherwise.

The *distance* $d(u, v)$ between two vertices u and v is the length of the shortest path which connects u and v . If there is a pair of vertices u, v with no path from u to v (or from v to u), $d(u, v) = \infty$, i.e., H is not connected.

A *connected component* is a maximal set of vertices $\mathcal{X} \subseteq \mathcal{V}$ such that, for all $u, v \in \mathcal{X}$, $d(u, v) \neq \infty$. The diameter $d(H)$ of H is defined as $d(H) = \max\{d(u, v) \mid u, v \in \mathcal{V}\}$. If H is disconnected, $d(H) = \infty$.

Figure 2.3 displays an example of such definitions.

It is interesting to point out that the notion of hyperedge incidence and vertex adjacency in a hypergraph is set-valued and quantitative, in the sense that two hyperedges can intersect at any number of vertices and two vertices can belong to any number of shared hyperedges. Otherwise stated, hypergraph walks have both length and width associated with them. Based on this consideration, Aksoy et al. [4] developed the

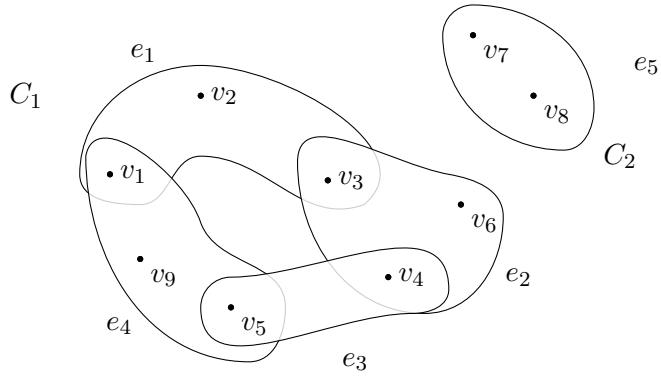


Figure 2.3: The above hypergraph has two connected components, C_1 and C_2. Hence, the diameter of H is $d(H) = \infty$, while $d(C_1) = 3$ and $d(C_2) = 1$. $P = v_9, e_4, v_5, e_3, v_4, e_2, v_3, e_1, v_2$ is a path from v_9 to v_2. The shortest path between v_9 and v_2 is $P' = v_9, e_4, v_1, e_1, v_2$. The distance $d(v_9, v_2) = 2$, corresponding to the length of P'. This hypergraph is simple.

concept of s-walk on a hypergraph, where s either controls for the size of edge intersection or the number of shared hyperedges. An s-walk (on hyperedges) is a sequence of distinct hyperedges in which each consecutive pair of hyperedges have at least s vertices in common. A formal definition follows:

For a positive integer s, an s-walk of length k and width s between hyperedges f and g is a sequence of hyperedges, $f = e_{i_0}, e_{i_1}, \ldots, e_{i_k} = g$, where $s \leq |e_{i_{j-1}} - e_{i_j}|$ for $j = 1, 2, \ldots, k$ and $i_{j-1} \neq i_j$.

On the dual hypergraph H^* (see Section 2.1.2), an s-walk (on vertices) is a sequence of adjacent vertices in which each successive pair of vertices belong to at least s shared hyperedges. Since in a graph a pair of vertices can belong to at most one edge, the usual graph walk between vertices u and v on a graph G is equivalent to a 1-walk between hyperedges u^*, y^* on the dual, G^*. Hence, when $s = 1$, a 1-walk corresponds to the usual graph walk.

The basic yet important properties of walks in graphs also extend to s-walks on hypergraphs. For instance, the existence of an s-walk between hyperedges defines an equivalence relation under which hyperedges can be partitioned into s-connected components. For an in-depth description of the notion of s-walk and their properties, we remand the reader to the work of Aksoy et al. [4].

2.1.5 Centrality Measures

Graph distance-based metrics can be trivially transferred to hypergraphs. Nonetheless, these metrics can also be generalized to accommodate the higher-order nature of those structures, as hypergraph walks have both length and width.

Based on their proposed notion of s-walk, Aksoy et al. [4] generalized the existing graph-distance based metrics to hypergraphs. For a hypergraph H , the s-distance between u and v is equivalent to the graph distance between u^* and v^* in the $s-$line graph $L_s(H)$ (see Section 2.2.2). Consequently, the following s-distance based measures are equivalent to

their graph counterparts on $L_s(H)$. Further, whenever H is a graph, they reduce to their graph counterparts on H^* for $s = 1$.

s-betweenness centrality. In a graph (2-uniform hypergraph), the betweenness centrality of a vertex v is the ratio of the number of non-trivial shortest paths between any pair of vertices in the graph that pass through v divided by the total number of non-trivial shortest paths in the graph. This measure quantifies the importance of a node in passing the information through the network. The betweenness is also defined for edges, counting the number of the shortest paths that go through an edge.

Formally, the s-betweenness centrality is defined as

$$C_B^s(v) = \sum_{\substack{x,y \in \mathcal{V} \setminus \{v\} \\ x \neq y}} \frac{\sigma_{xy}^s(v)}{\sigma_{xy}^s},$$

where $\sigma_{xy}^s(v)$ is the number of the s-node(edge)-shortest-paths between two vertices x and y that pass through v, while σ_{xy}^s is the total number of s-node(edge)-shortest-paths between x and y.

s-closeness centrality. In a connected component, the normalized closeness centrality of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. Formally, the s-closeness centrality is defined as

$$C_C^s(v) = \frac{|\mathcal{V}| - 1}{\sum_{u \in \mathcal{V} \setminus \{v\}} d_s(u, v)},$$

where $\mathcal{V}$ is the set of vertices in the s-line graph, and d_s is the s-shortest path distance.

Once again, we remind the reader the work of Aksoy et al. [4] for more details.

It is worth noting that also the definition of degree centrality for vertices in a hypergraph straightforwardly extends the same notion defined for graphs: the degree centrality of a node is given by the number of hyperedges it is contained in.

The definition of s-betweenness centrality

We independently developed the notion of s-betweenness centrality in [14].

2.2 Graphs versus Hypergraphs

In the previous sections, we saw how hypergraphs generalize standard graphs by defining (hyper)edges among multiple vertices instead of only two elements. Here, after recalling some graph definitions, we will describe how hypergraphs can be transformed into a possible corresponding graph representation.

2.2.1 Recalling Graphs

Definition 2.2 : Simple Graph

A *simple graph* $G = (\mathcal{V}, \mathcal{E})$ consists of a non-empty set $\mathcal{V}$ representing vertices, and a set $\mathcal{E}$ of unordered pairs of elements of $\mathcal{V}$ representing edges. A simple graph has no arrows, no loops, and cannot have multiple edges joining vertices.

When multiple edges between nodes are permitted, we refer to such structure with the term *multigraph*.

Definition 2.3 : Multigraph

A *multigraph* $G = (\mathcal{V}, \mathcal{E}, f)$ consists of a non-empty set $\mathcal{V}$ representing vertices, a set $\mathcal{E}$ of unordered pairs of elements of $\mathcal{V}$ representing edges, and a function f, which assigns to each edge an unordered pair of endpoint nodes, defined as follows:

$$f : \mathcal{E} \to \{\{u, v\} : u, v \in V \text{ and } u \neq v\}.$$

If $e_1, e_2 \in \mathcal{E}$ are such that $f(e_1) = f(e_2)$, then we say e_1 and e_2 are multiple or parallel edges.

A 2-uniform hypergraph is a simple graph if it has no repeated hyperedges. A hypergraph whose rank is at most 2 allows repeated hyperedges of size 2, but also hyperedges of size 1 (i.e., loops).

2.2.2 Hypergraphs to Graph Representations

In the literature, hypergraphs have been usually converted into a corresponding graph representation, especially for computation convenience and the easiness of dealing with graphs rather than higher-order structures. We can transform a hypergraph into one of the following graph representations: (*i*) line graph, (*ii*) two-section graph, and (*iii*) bipartite graph. It is worth highlighting that such transformations may bring a loss of information regarding the original hypergraph structure. In the following, we will briefly describe each representation. For a description of their properties, we refer the reader to "Hypergraph Theory: An Introduction" [47].

Line Graphs

Definition 2.4 : Line Graph

Let $H = (\mathcal{V}, E = (e_i)_{i \in \mathcal{I}})$ be a hypergraph. The *line graph* (or *representative graph* or *intersection graph*) of H is the graph $L(H) = (\mathcal{V}', \mathcal{E}')$ such that:

- $\mathcal{V}' = \mathcal{I}$ or $\mathcal{V}' = E$ when H is without repeated hyperedge;
- $\{i, j\} \in \mathcal{E}', i \neq j \iff e_i \cap e_j \neq \emptyset$.

Line graphs may also be defined with additional edge weights where the edge $\{i, j\} \in \mathcal{E}', i \neq j$ has weight $|e_i \cap e_j|$. By definition of matrix multiplication, the line graph of a hypergraph with incidence matrix $\mathbf{H}$ has edge-weighted adjacency matrix $\mathbf{H}^\intercal\mathbf{H}$ with diagonal entries equal to 0.

The definition of line graphs can be generalized to $s-$ line graphs [4] (denoted with $L_s(H)$), where each vertex represents a hyperedge with at least s vertices in the hypergraph, and two vertices are linked in the $s-$line graph if their corresponding hyperedges intersect in at least s vertices in the hypergraph. Formally, $\{i, j\} \in \mathcal{E}', i \neq j \iff |e_i \cap e_j| \geq s, |i| \geq s, |j| \geq s$.

Figure 2.4 illustrates an example of line graph.

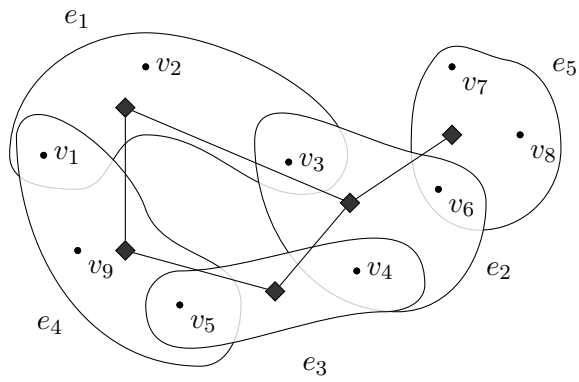


Figure 2.4: The figure shows a hypergraph $H = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_i\}_{i\in[1,9]}$ and $E = \{e_j\}_{j\in[1,5]}$, and its line graph $L(H) = (\mathcal{V}', \mathcal{E}')$. The vertices of $L(H)$ are the black diamonds and its edges are the lines between the diamonds.

Limitations

- Distinct hypergraphs can have identical line graphs as they lose the information about the composition of each hyperedge, storing only whether two hyperedges intersect but not in which manner;
- Sparse hypergraphs can yield relatively dense line graphs as a vertex of degree d in the hypergraph yields $\binom{d}{2}$ edges in its line graph.

Two-section Graphs

Definition 2.5 : Two-section graph

Let $H = (\mathcal{V}, E)$ be a hypergraph. The *two-section* or *clique graph* of H is the graph, denoted with $[H]_2$, whose vertices are the vertices of H and where two distinct vertices form an edge if and only if they are in the same hyperedge of H . In other words, each hyperedge of H appears as a complete sub-graph in $[H]_2$.

Figure 2.5 shows an example of this representation.

The *generalized two-section* of H, denoted with $G[H]_2$, is the *labeled-edge multigraph* whose vertices are the vertices of H and where the vertices u and v are connected by an edge, labeled with e, if $\{u, v\} \subseteq e$.

The *weighted two-section* of H, denoted with $[H]_2^w = (\mathcal{V}, \mathcal{E}', \mathbf{W})$, where $\mathbf{W} \in \mathbb{R}^{n \times n}$, is the weighted clique graph in which the weight $w(e) =$

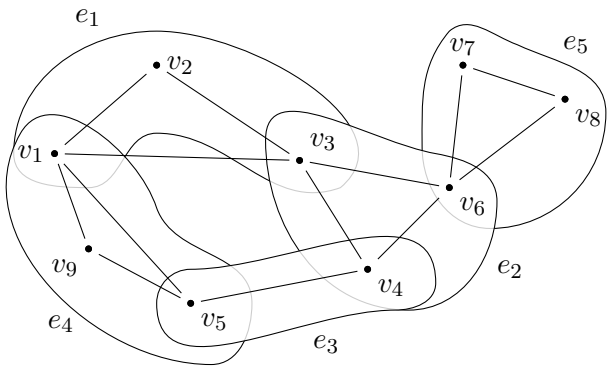


Figure 2.5: The figure shows a hypergraph $H = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_i\}_{i \in [1,9]}$ and $E = \{e_j\}_{j \in [1,5]}$, and its clique graph $[H]_2 = (\mathcal{V}, \mathcal{E}')$. The edges of $[H]_2$ are the lines between the vertices of H.

$\mathbf{W}(u, v)$ of an edge $e = \{u, v\} \in \mathcal{E}'$ corresponds to the number of hyperedges containing both u and v, i.e., $w(e) = \mathbf{W}(u, v) = \mathbf{W}(v, u) = |H(u) \cap H(v)|$.

Limitations

- Clique graphs completely lose the notion of groups since pairwise connections substitute each high-order interaction. Hence, it is impossible to recover the original hypergraph structure from the two-section graph (being a simple graph), unless we use the generalized two-section and, thus, the more powerful model of a labeled-edge multigraph.
- Each hyperedge of size k is transformed into $\frac{k \times (k-1)}{2}$ edges.

Incidence Graphs

Definition 2.6 : Incidence graph

Let $H = (\mathcal{V}, E)$ be a hypergraph. The *incidence graph* of H is the bipartite graph $I(H) = (\mathcal{V}', \mathcal{E}')$ with a vertex set $\mathcal{V}' = \mathcal{V} \cup E$, and where $v \in \mathcal{V}$ and $e \in E$ are adjacent if and only if $v \in e$.

Figure 2.6 illustrates the incidence graph associated with a hypergraph.

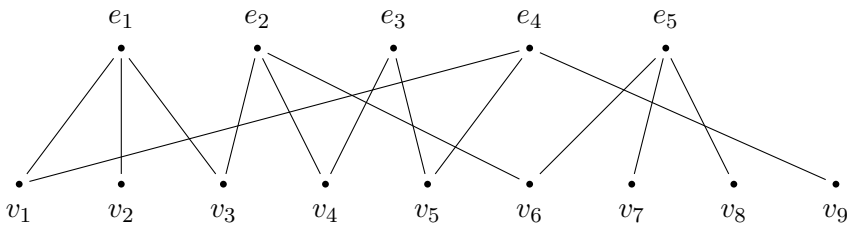


Figure 2.6: The figure shows the incidence graph $I(H) = (\mathcal{V}', \mathcal{E}')$ of the hypergraph H, with a vertex set $\mathcal{V}' = \{v_i\}_{i\in[1,9]} \cup \{e_j\}_{j\in[1,5]}$. The edges of $I(H)$ are the lines between the vertices $\mathcal{V}'$.

Limitations

Vertices in the original system do not interact directly anymore as the interaction layer always mediates their relationship. This interaction structure implies that any measure or dynamic process defined on the bipartite representation must consider this additional complexity.

2.3 Notation Summary

This section summarizes the notation and the concepts introduced in this chapter and used throughout the thesis. While Table 2.2 specifically lists the mathematical notation, Table 2.3 focuses only on the hypergraph-related concepts that will be explicitly referred to in the remainder of this dissertation.

Table 2.2: Summary of the notation used throughout this thesis.

Symbol	Interpretation
$\mathbf{v}$	Bold-faced lower case letters are used to identify vectors.
$\mathbf{v}_i$	The indexed notation identifies the i-th element of the vector $\mathbf{v}$.
$\mathbf{A}$	Bold-faced capital letters are used to identify matrices.
$\mathbf{A}(i, j)$	The indexed notation identifies the j-th element in the i-th row of the matrix $\mathbf{A}$.
$\mathcal{A}$	Calligraphic capital letters are used to identify sets.
$\lvert\mathcal{A}\rvert$	The cardinality (i.e., number of elements) of the set $\mathcal{A}$.
$[\![P]\!]$	Indicator function: it equals 1 if the predicate P is true; 0 otherwise.

An exception to the default notation

Although hyperedges are technically sets, they are identified with lower case letters. Hence, we focus the attention on hyperedges as *elements* of the family of hyperedges E, and not as sets per se.

Table 2.3: Summary of the notation used throughout this thesis.

Symbol/Concept	Interpretation
H, G	Capital letters are used to identify hypergraphs and graphs.
$\mathcal{V}$	Vertex set, where $\lvert\mathcal{V}\rvert = n$.
$E = (e_i)_{i \in \mathcal{I}}$	Family of hyperedges over a finite set of indexes $\mathcal{I}$, with $\lvert E\rvert = m$.
u, v, e	Lower case letters identifies nodes and (hyper)edges.
$\mathbf{H}(v_i, e_j)$	Incidence matrix $\mathbf{H}$ indicating whether the vertex v_i is in the hyperedge e_j.
$\lambda_e(v)$	Weight of the vertex v in the hyperedge e.
$H(v)$	Family of hyperedges containing v.
$\deg(v)$	Degree of a vertex v.
$\kappa(e)$	Degree or cardinality of a hyperedge e.
d-regular	Hypergraph in which each vertex has degree d.
k-uniform	Hypergraph in which each hyperedge has cardinality k.
H^*	Dual hypergraph H^* of H.
$[H]_2$	Two-section or clique graph of a hypergraph H.
$[H]_2^w$	Weighted two-section of a hypergraph H.
$I(H)$	Incidence graph of a hypergraph H.

3

Coding Hypergraphs

> Everybody should learn to program a computer, because it teaches you how to think.
>
> Steve Jobs

In short

This chapter provides an overview of the available software technologies to manipulate and explore hypergraphs. Specifically, the first part of the chapter reports the main software libraries to analyze these structures, describing their peculiarities, pros, and cons (see Section 3.1). The second part of the chapter focuses on the Julia programming language, describing its primary characteristics and performance (see Section 3.2).

3.1 Hypergraph Software Frameworks

Hypergraphs are the natural representation of many real-world systems. However, the powerful expressiveness of such mathematical structures has a few drawbacks: dealing with the inherent complexity of higher-order relationships and the lack of appropriate tools and algorithms for their study. For this reason, hypergraphs have been little used in the literature in favor of their graph-counterpart (see Section 2.2). In turn, this trend caused the reduced development of software frameworks suitable

for modeling and mining these structures. Recently, a shift in this research paradigm happened, also thanks to the newly available computational capabilities (e.g., Cloud platforms). Over the last few years, a growing body of literature has been conveying its efforts to investigate hypergraphs to design more effective solutions in various domains [32]. Hypergraphs have, thus, risen to prominence, giving a fresh impetus to the development of hypergraph software frameworks.

In the following, we list some state-of-art software libraries in alphabetical order, focusing on the availability of their code and their capability to model and analyze hypergraphs.

Chapel HyperGraph Library (CHGL) [104] is a library for hypergraph computation, written in the emerging Chapel parallel programming language. Being developed by the Pacific Northwest National Laboratory (PNNL) since 2018, the library is released under the MIT license and publicly available in a GitHub open-source repository [105]. The main idea behind this library was to provide a High-Performance Computing (HPC) -class computation with high-level abstractions and modern language support for parallel computing on shared and distributed memory systems.

CHGL primarily focuses on hypergraph generative models while providing a high- ly abstract interface for implementing hypergraph algorithms. The library has a single main hypergraph data structure, storing the hypergraph's adjacency list, which supports both static and dynamically growing hypergraphs thanks to the intrinsic characteristics of the language. In more detail, the hypergraph is stored using two (possibly distributed) arrays, one for vertices and one for hyperedges. Each vertex (resp. hyperedge) further refers to a non-distributed array representing the hyperedges it is contained in (resp. the vertices it contains). Every inclusion is stored in two directions, both at the included vertex and the including hyperedge, resulting in storing the hypergraph and its dual.

CHGL is still in its early development stage, as well as the associ-

ated documentation. The latest release of the library (29 Oct 2020) includes an efficient homology computation functionality and the computation of the s-distance between two nodes in the hypergraph.

Gspbox [132] is a Matlab toolbox that performs a wide variety of operations on a graph. The software is free, and it is released under the GNU General Public License (GPLv3). Gspbox is based on spectral graph theory, and many of the implemented features can scale to very large graphs. Gspbox supports hypergraphs modeling, including weighted hyperedges and vertices with coordinates in the space. However, hypergraph manipulation is obtained by representing the model as a graph. For this reason, although all graph functionalities are available, the library does not provide any specific solutions or optimization for hypergraphs.

A very recent version of this library is also available in Python. Nonetheless, no hypergraph support is implemented at all.

Halp [139] is a Python software package providing both directed and undirected hypergraph implementations and a few major and classical algorithms. The library was developed by Murali's Research Group at Virginia Tech and released under the GPL-3.0 license.

Halp provides several statistics on the hypergraph structure (e.g., mean in/out-degree) and a module, integrated with the Python library NetworkX, to transform a hypergraph into the corresponding clique graph. Halp also implements random walk and shortest-paths functionalities.

Hygra [164] is a suite of efficient parallel hypergraph algorithms written in C/C++. It includes algorithms for betweenness centrality, maximal independent set, k-core decomposition, hypertrees, hyperpaths, connected components, PageRank, and single-source shortest paths. This framework is released under the MIT license and integrated within the Ligra package (A Lightweight Graph Processing Framework for Shared Memory [165]).

Hygra uses the incidence graph representation of a hypergraph to store the high-order structure and Ligra's data structures for representing subsets of vertices and hyperedges as well as operators for mapping application-specific functions over these elements. Further, the framework separates the operations on vertices from operations on hyperedges; thus, it requires a careful definition of the functions for vertices and hyperedges to preserve correctness. Hygra also inherits from Ligra various optimizations developed for graphs, including switching between different traversal strategies, edge-aware parallelization, bucketing for prioritizing the processing of vertices, and compression.

Hypergraph [66] is a newly born Rust data structure library released under the MIT license. Currently, this library only allows generating directed hypergraphs; however, the peculiarities of the Rust language could open to fast parallel implementation of the most common network algorithms as future development.

HyperGraphLib [126] is a C++ library for hypergraphs modeling, released under the MIT license. This library only provides basic functionalities for generating hypergraphs having specific characteristics, like d-regular, k-uniform, simple, and linear hypergraphs, isomorphism functionalities, and path-finding algorithms.

HyperNetX [110] is probably the most comprehensive and updated Python package to model and analyze hypergraphs. As CHGL, HyperNetX has been developed by the PNNL since 2018. The library is released under the 3-Clause BSD license and is publicly available on a GitHub repository [155].

HyperNetX generalizes traditional graph metrics to hypergraphs (e.g., s-paths [4]), and provides hypergraph-specific algorithms. The library offers functionalities to generate hypergraphs, evaluate centrality measures, and compute clusters. HyperNetX further supports the bipartite representation of a hypergraph, along with the

possibility to load hypergraphs from their bipartite view. In addition, the library exports some visualization functionalities based on Euler diagrams.

The library's latest version (13 August 2021) also supports a contagion module to study SIS and SIR contagion networks using hypergraphs and an add-on for providing optimized C++ implementations of many of the available hypergraphs methods. All code is well-documented and accompanied by several comprehensive tutorials.

HyperX [108] is a general-purpose distributed hypergraph processing framework built on top of Apache Spark, developed in 2015 and implemented using the Scala programming language. HyperX is available as an open-source repository on GitHub [107]; however, no documentation is available, and the last update dates back to 2015.

HyperX sets itself as the corresponding GraphX for hypergraphs, supporting the same programming model but providing native support for elaborating higher-order structures. Directly processing hypergraph data, HyperX obtains significant speedup with respect to using a hypergraph to graph transformation and then exploiting GraphX APIs.

HyperX includes algorithms for random walks, label propagation, and spectral learning. In particular, partitioning algorithms assume greater importance in such a context as they are used to load-balancing the work of each node with low communication costs among the workers. HyperX stores hypergraphs using fixed-length tuples containing vertex and hyperedge identifiers, and the algorithms are written using hyperedge programs and vertex programs that are iteratively applied on hyperedges and vertices, respectively.

Iper [51] is a JavaScript library for hypergraphs, released under the MIT license. The library defines the hypergraph structure and allows the

user to define meta-information for vertices. However, Iper does not implement any hypergraph-specific algorithm or integrate with other graph libraries for classical statistics and algorithms.

NetworkR [168] is an R package, distributed under the MIT license, with a set of functions for analyzing social and economic networks, including hypergraphs. Being focused on graphs, the library does not offer any specific hypergraph-related functionality and only provides hypergraphs projection into graphs.

Multihypergraph [114] is a Python package for graphs, released under the GPL license. The library emphasizes the mathematical understanding of graphs rather than the algorithmic efficiency, and it only provides support for hyper-edges, multi-edges, and looped-edges. Specifically, this library implements only the graph memory model definition and isomorphism functionalities without defining any oth- er functionality and algorithm for graphs and hypergraphs.

Overall, we can identify two main driving factors guiding the development of each software framework, namely analytical comprehensiveness and HPC compliance.

In the first group, we can find all the libraries supporting a few to a wide range of hypergraph-specific functionalities. HyperNetX undoubtedly stands out among those thanks to the heavy contribution to its development over the last few years. Being HPC-specific, frameworks such as CHGL, Hygra, and HyperX belong to the second group. These libraries offer specific support for running heavy computations via parallel and distributed algorithms, allowing the possibility to scale to hypergraphs with millions of vertices and hyperedges. In particular, CHGL provides functions for accessing properties of hypergraphs, but its interface is much lower-level than the abstractions in HyperX and Hygra. Based on the comparison made by the authors of Hygra [164], their framework achieves significantly faster results thanks to the optimization made for frontier-based hypergraph algorithms. However, it is worth empha-

sizing that these tools require specific skills in working on parallel and distributed memory models.

Many other libraries, specifically designed to model and analyze graphs, offer little support to hypergraphs, often only implementing hypergraph models and graph transformations.

In general, all the listed libraries are a compromise between efficiency, which characterizes low-level languages such as C/C++, and the ease-of-use and expressiveness, peculiar to interpreted and scripting languages such as Python and R. With its latest release in August 2021, HyperNetX represents an exception to such a trade-off, offering a C++ optimization module callable through Python.

3.2 The Julia Programming Language

Julia [39] is a general-purpose dynamic programming language appropriate for scientific and numerical computing, with performance comparable to traditional statically-typed languages. Julia represents a single environment that is productive enough for prototyping and efficient enough for deploying performance-intensive applications.

Born in 2009 from Bezanson, Karpinski, Shah, and Edelman's work and officially launched in 2012, Julia was born to accommodate a growing problem in the scientific community: the need for a free language that was both high-level and fast. Hence, Julia was created to answer the *two language problem of data science* [48], where one programming language is used as glue to another language that actually implements compute-intensive algorithms. That is a typical workflow when coding and prototyping with a high-level language like R and Python, as the performance-critical parts have to be rewritten in C/C++ for performance. This double-step is highly inefficient because it introduces human error and wasted effort and slows time to market.

Figure 3.1 reports compiler performance on a range of common code patterns, such as function calls, string parsing, sorting, numerical loops, random number generation, recursion, and array operations. The vertical

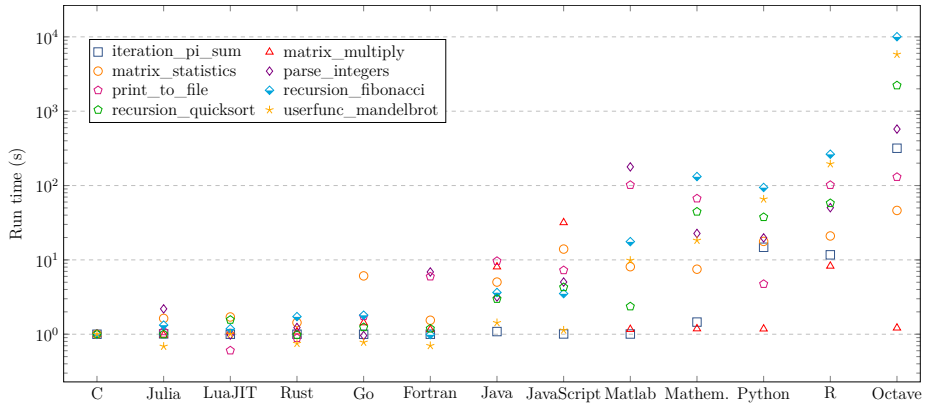


Figure 3.1: Run time of the most used programming languages for scientific computing over eight benchmark tasks. Run times are normalized against the C implementation. *Mathem.* stands for Mathematica.

axis shows each benchmark time normalized against the C implementation. We can note from the picture that the most suitable languages for implementing computing-intensive algorithms are C, Julia, LuaJIT, Rust, Go, and Fortran. As expected, the run time of the most used scripting languages, i.e., Python and R, are much higher than C. Details on this micro-benchmark are reported on the Julia website [112].

Figure 3.2 shows the average run time of each language evaluated over all benchmark programs, plotted versus the average length of their source files [180]. The vertical axis shows each benchmark time normalized against the C implementation. The ideal is having concise programs with short run times, i.e., the lower, the better on both axes. The plot shows the classic trade-off in technical computing. Compiled languages like C and Fortran (lower right) are fast to run but slow to write, while scripting languages like R and Python (upper left) are fast to write but slow to execute. The Julia language stands out in the overall picture, achieving the lowest average run time and code length.

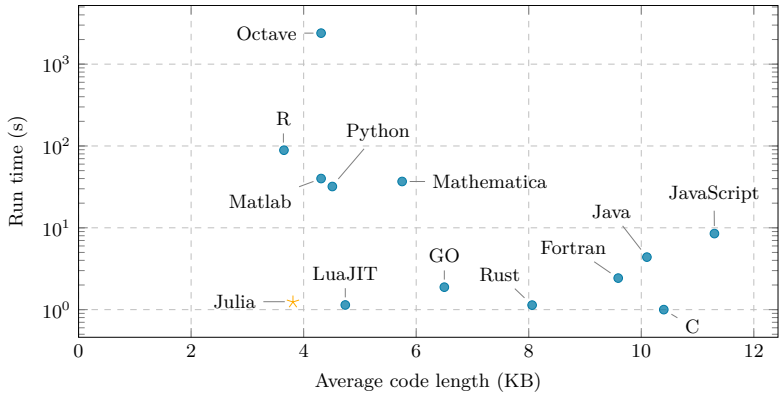


Figure 3.2: Average performance vs. code length.

3.2.1 Key Features

Julia's main web page lists six principal characteristics of the language. Julia is

Fast and Dynamic. This language was designed to achieve high performance. For this reason, Julia programs are *compiled* to optimized native machine code for multiple platforms via LLVM. Thus, like C and Fortran, Julia is compiled. However, unlike those programming languages which are compiled before execution (*ahead of time*), Julia is compiled at runtime (*just in time* for execution). This design choice allows Julia to be *dynamically typed*, thus looking more like a scripting language like Python.

The dynamic type system nicely works thanks to the Julia type inference process. For each Julia function, it determines bounds for the types of each of its variables, as well as bounds on the type of the return value from the function itself. In this manner, Julia compiles a native version of a function the first time it is run with a specific set of argument types, caching its bytecode. If the same function - but with different argument types - is called, Julia recompiles for the given types, caching the new bytecode in another location. Subse-

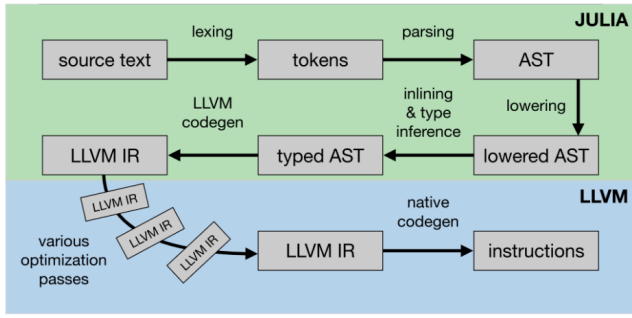


Figure 3.3: The Julia compiler, from source to machine code. Source: `http://slides.com/valentinchuravy/julia-parallelism`.

quent runs use the appropriate bytecode with recompilation.

In a few words, after the typed code has been generated, it gets compiled to LLVM IR (Intermediate Representation). Then the IR code passes through LLVM, which applies several optimization phases to generate fast native code, finally getting executed. The generated code from each compilation phase can be inspected through Julia-specific macros. Figure 3.3 shows the whole compilation process.

Composable. Julia uses multiple dispatch as a paradigm, making it easy to express object-oriented and functional programming patterns.

The notion of *dispatch* refers to the choice of which method to execute when a function is applied. Julia allows the dispatch process to choose which function's methods to call based on the number of arguments given and the types of all of the function's arguments. Using all of a function's arguments to choose which method should be invoked, rather than just the first as happens in traditional object-oriented languages, is known as *multiple dispatch*.

General. Julia provides asynchronous I/O, metaprogramming, debugging, logging, profiling, and a package manager, among other functionalities.

In particular, *metaprogramming* support is the strongest legacy of Lisp in the Julia language. Like Lisp, Julia represents its own code as a data structure of the language itself. Since code is represented by objects created and manipulated within the language, a program can transform and generate its own code. That allows sophisticated code generation without extra build steps and true Lisp-style macros operating at the level of abstract syntax trees. In contrast, preprocessor "macro" systems, like that of C and C++, perform textual manipulation and substitution before any actual parsing or interpretation occurs. Because Julia data structures represent all data types and code in Julia, powerful reflection capabilities are available to explore the internals of a program and its types just like any other data.

Reproducible. Julia offers the possibility to create environments to enhance code reproducibility. For instance, by checking a project environment into version control (e.g., a git repository) along with the rest of the project's source code, one can reproduce the exact state of the project and all of its dependencies. The manifest file, in particular, captures the exact version of every dependency, identified by a cryptographic hash of its source tree, which makes it possible for the package manager to retrieve the correct versions and be sure that you are running the exact code that was recorded for all dependencies.

Open source. Julia is an open-source project with over 1,000 contributors. It is distributed under the MIT license, and its source code is available on GitHub [111].

In a nutshell, Julia is a multi-paradigm language that combines imperative, functional, and object-oriented programming features. Julia provides ease and expressiveness for high-level numerical computing, in the same way as languages such as R, MATLAB, and Python, but also supports general programming. To achieve this, Julia builds upon the lineage

of mathematical programming languages and borrows much from popular dynamic languages, including Lisp, Perl, Python, Lua, and Ruby.

4

The Social Influence Diffusion Problem

> That all our knowledge begins with experience there can be no doubt.
>
> Immanuel Kant

In short

This chapter presents current literature related to the analysis of social influence phenomena abstracted and studied via the lens of networks. We specifically focus on discussing the social influence maximization (SIM) problem, as the contribution of this dissertation relates to a variant of the problem just mentioned. In Section 4.1, we first introduce some basic concepts related to influence diffusion models. We then briefly outline the origin of the SIM problem and the Target Set Selection variant in the context of graphs in Section 4.2. In Section 4.3, we finally concentrate on current literature on social influence diffusion analyzed through hypergraphs.

4.1 Influence Diffusion Models

Social influence occurs due to the diffusion of information in the network. Several diffusion models have been proposed to capture how the spreading process happens, and there exists a vast amount of literature on designing diffusion models in the areas of data mining, databases, networks, and epidemiology [129]. Each model applies different mechanisms to represent how a user switches its status from inactive to active; in other words, how a vertex can be influenced by its neighbors. Generally, all diffusion models abstract the network to study with a graph, where vertices are the users of the social networks and edges the social ties between pairs of users. Each vertex has an associated diffusion threshold, measuring the hardness of influencing the user given in numerical scale (the higher the value, the harder it is to influence the user). The influence probability between two users is usually modeled as edge weight, instead.

All diffusion problems are formalized based on a specific spreading model. Two popular diffusion models commonly used to study the influence maximization problem are the (i) Linear Threshold model, which captures the collective behavior of the agents [26], and the (ii) Independent Cascade model, which collects the independent behavior of the agents. A brief description of both models follows.

Linear Threshold model (LT). The LT model was introduced by Granovetter [87] and Schelling [161] in 1978. The underlying idea of the LT model is that a user can switch its status from inactive to active if a sufficient number of its incoming neighbors are already influenced.

Formally, each edge $e = (u, v) \in \mathcal{E}$ has a weight $w(e)$, and each vertex v has a diffusion threshold Θ_v. Let $\mathcal{N}(v)$ be the set of incoming neighbors of the vertex v, satisfying $\sum_{u \in \mathcal{N}(v)} w(e) \leq 1$. Let us consider an instance of the LT diffusion process, which proceeds in discrete steps. Initially, all vertices in the initial set of influenced users are activated, while the others remain inactive. Then, at each iteration t, all users that were active in the previous step $t-1$ remain active, and any user v that was inactive in $t-1$ switches to active if

the total weight of its active neighbors in $\mathcal{N}(v)$ is at least Θ_v. The diffusion process terminates when no more users can be activated.

Independent Cascade model (*IC*). The IC model is a classic and well-studied diffusion model introduced by Goldenberg [83] in 2001. The model introduces an influence probability $p_{u,v}$ to each edge $e = (u, v) \in \mathcal{E}$: each vertex u may independently influence each neighbor v with probability $p_{u,v}$.

As in the LT model, the IC model unfolds in discrete steps. Initially, all vertices in the initial set of influenced users are activated, while the others remain inactive. Then, at each iteration t, every active user in the previous step $t - 1$ will try to activate each outgoing neighbor v (inactive in step $t - 1$) with probability $p_{u,v}$. The vertex u has only one chance to activate its outgoing neighbors: after that, u remains active and stops the neighbors' activation task. The diffusion process terminates when no more vertices can be activated.

As stated at the beginning of this paragraph, there exist a plethora of diffusion models. For a detailed and exhaustive treatment of those, we refer the reader to [197].

4.2 SIM Problem in Graphs

Initially introduced by Domingos and Richardson [65] in the context of viral marketing, the Social Influence Maximization (SIM) problem has been studied since the early two thousands. In this section, we concisely outline the problem and the current research landscape. We refer the reader to three recent surveys giving a thorough description of the influence maximization problem and its variants, solution methodologies, and real-life applications [26, 129, 152].

Kempe et al. [117] first modeled influence maximization as a combinatorial optimization problem in 2003 and investigated its computational complexity under the LT and IC diffusion models with randomly chosen thresholds. The problem studies a social network abstracted as a graph

$G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (i.e., users) and $\mathcal{E}$ is the set of (directed/undirected) edges (i.e., social links between users). The goal is to find a k-sized set of users with the maximum influence in G. Kempe et al. proved the SIM problem to be NP-hard to approximate within a factor of $O(n^{(1-\epsilon)})$ for all $\epsilon > 0$.

In its basic version, the Target Set Selection (TSS) problem, proposed by Ackerman et al. [1], adds another parameter to the SIM problem, asking to find out a subset of at most k vertices, such that after the diffusion process is over at least λ vertices are influenced. The TSS Problem is a more generalized version of many standard graph-theoretic problems [26], such as dominating set with threshold [90] and vertex cover [53] (in this problem, vertex threshold is set equal to the number of neighbors of the vertex). Chen [53] studied the TSS problem with fixed arbitrary thresholds and proved a strong inapproximability result that makes unlikely the existence of an algorithm for the TSS problem on graphs with an approximation factor better than $O(2^{\log^{1-\epsilon}|V|})$. Recently, Cordasco et al. [58] presented an algorithm for the TSS problems on graphs, which provides an optimal solution (i.e., a minimum size subset of vertices that influence the whole network) if the network is either a tree, a cycle, or a complete graph.

Current research trends focus on scalability issues, computing realistic diffusion probabilities, and considering benefits and costs as another component in the SIM problem [26]. In particular, most of the work in the last years focused on reducing the time complexity of the algorithmic solutions, given the NP-hardness nature of the SIM problem under both the LT and IC models. Current state-of-the-art methods are represented by the following algorithms [95]: CELF [125], IMM [176], SSA and D-SSA [144]. Other lines of inquiry in the context of social influence diffusion are understanding how diffusion mechanics differ across social media and modeling different types of influence within a network. Kim et al. [118] moved towards the first direction, exploring news diffusion instances across different social media platforms and finding that influence between different media types is varied by the context of infor-

mation. Li et al. [128] followed the second direction, formally defining a multiple influences diffusion model by considering the influential relationships and individual's personalized traits, such as interests and trusts. Recently, Chathurani et al. [162] put together these two aspects, investigating several influence patterns exercised by different users within and across online social media platforms.

4.3 SIM Problem in Hypergraphs

Little literature exists on the SIM problem on hypergraphs as these structures rose to prominence only recently in the academic and industry landscape.

Hypergraphs first appeared in the context of SIM in the work of Borgs et al. [44] in 2014, even though the study's goal was to efficiently maximize the expected influence under the standard IC model in a network modeled via a directed edge-weighted graph. The idea underlying the algorithm proposed by the authors is based on a polling process, which selects a vertex v uniformly at random and then determines the set of vertices that would have influenced v. Intuitively, if the process is repeated multiple times, and a specific vertex u often appears as an influencer, it is likely for u to be one of the most influential vertices. The algorithm proposed by Borgs et al. proceeds in two steps. According to the described random sampling technique, the first phase generates a sparse hypergraph representation of the input graph. Each hyperedge corresponds to a set of individuals influenced by a randomly selected vertex in the transposed graph. Hence, the hypergraph represents an influence estimation: given a set of vertices S, the total degree of S in the hypergraph is approximately proportional to the influence of S in the original graph. In the second phase, the algorithm performs a greedy procedure that repeatedly chooses the vertex with the highest degree in the hypergraph and then removes the chosen vertex and all incident hyperedges. The selected k vertices represent the final seed set of size k. Borgs et al. proved the algorithm being near-optimal, with a lower bound of $\Omega(n + m)$.

In 2016, Gangal et al. [78] first dealt with the problem of influence maximization directly on hypergraphs. The authors extended the notion of a vertex to an affiliation or group (represented by a hyperedge) being influenced, considering a hyperedge activated if the majority of its vertices are influenced at the end of the diffusion process. The primary objective of this work is to maximize the number of hyperedges influenced rather than the number of vertices influenced (HEMI problem). Based on this idea, Gangal et al. extended the IC model to hypergraphs. They modeled the diffusion process on a hypergraph H via its directed incidence graph $I(H)$ to have independent influence probabilities flowing in either direction. In this model, a vertex v active at the time step $t-1$ can influence an incident hyperedge e with probability $p_{v,e}$ in the following time step t through the edge (v, e). Similarly, a hyperedge e active in $t-1$ can influence an incident vertex v with probability $p_{e,v}$ during t through the edge (e, v). Hence, hyperedges act as carriers of the influence, allowing it to flow from one vertex to another through them. However, according to the HEMI objective, a hyperedge is included in the final seed set only if most of its vertices are influenced, regardless of whether it has acted as a carrier. Gangal et al. proved the HEMI problem to be non-submodular under the diffusion model proposed, and they indicated as future work devising an algorithm that can solve the problem with some provable guarantee.

In 2019, Zhu et al. [198] revised the SIM problem under a group influence perspective, based on a similar intuition. Resembling real-life dynamics, the authors formalized the Group IM (GIM) problem, whose goal is selecting k seed users such that the number of eventually influenced groups is maximized. The proposed formulation strictly resembles the problem addressed by Gangal et al.; nonetheless, we can note a few differences in the problem setting. First, in Zhu et al., a group is influenced if at least a fixed percentage β of users in it are influenced under the classical IC model. Conversely, Gangal et al. fixed a majority threshold on groups and considered a hyperedge activated only if most of its vertices are influenced at the end of the diffusion process. Further, Gangal et al. extended the IC model to hypergraphs, considering vertices influencing

hyperedges and vice-versa. Second, Zhu et al. consider a directed hypergraph while Gangal et al. an undirected hyper-network. After analyzing the complexity and approximability of the GIM problem, proved to be NP-hard, Zhu et al. developed a group coverage maximization algorithm and proposed a sandwich approximation framework for the problem, formulating a lower bound and upper bound of the objective function.

The same research group proposed two other works related to the study of information diffusion in social networks abstracted as a directed hypergraph, focusing specifically on crowd influence. In [200], Zhu et al. modeled the crowd influence as a hyperedge $e = (H_e, v)$ with weight $0 \leq p_{H_e,v} \leq 1$, where H_e is the head vertex-set and v is the tail vertex, meaning that v will be influenced by H_e with probability $p_{H_e,v}$ only after each vertex in H_e is influenced. Their proposed objective function is selecting k initially-influenced seed users in a directed hypergraph $G = (\mathcal{V}, \mathcal{E}, \mathbf{P})$ to maximize the expected number of eventually-influenced users. Zhu et al. proved the problem to be NP-hard under the IC model and proposed an algorithm preserving a $(1 - 1/e - \epsilon)$-approximation, based on the D-SSA [144] method for graphs to obtain seed vertices.

In [199], Zhu et al. continued analyzing the Composed Influence Maximization (CIM) - proposed in [200], considering social networks modeled as a directed hypergraph. In this work, the authors focused on developing lower and upper bounds of the CIM problem's objective function to then design a greedy strategy based on the lower bound maximization for solving it. Zhu et al. also formulated a sandwich approximation framework, preserving the theoretical analysis result.

Recently, Wang et al. [185] proposed to add different types of relations in the SIM setting. Specifically, when analyzing diffusion processes, they consider both user-to-user friendship relations and relationships generated by online activities (for instance, if two users commented on the same post). The intuition behind this idea is based on the fact that many online social network users may participate in multiple common online activities and influence each other even though they are not friends. Wang et al. abstract the so-generated heterogeneous network with a hypergraph

$H(\mathcal{V}, \mathcal{E}, \mathcal{E}_1, \ldots, \mathcal{E}_l)$, where $\mathcal{V}$ denotes the set of users, $\mathcal{E}$ denotes the user-user links, and $\mathcal{E}_i$, $i \in [1, l]$, denotes the set of hyperedges of type i, in which each hyperedge is a set of users who participated in the same online activity. In the diffusion model proposed by Wang et al., influence spreads from one user to another according to three criteria: (i) direct user-to-user influence, (ii) user-activity-user influence, and (iii) composite influence (user-to-user and user-activity-user) by one-hop neighbors. The authors then approximate the vertices' influence by defining an influence centrality based on a random walk on hypergraphs. In this manner, they approximate the SIM problem by solving a centrality maximization problem. Wang et al. finally employ the Monte Carlo framework to estimate influence centrality and develop a greedy-based algorithm to solve the SIM problem under the IC and LT models.

Influence diffusion models are usually treated as specific cases of epidemic models [152]. The following two works are an example of such methodology applied to hypergraphs. We will discuss these articles also in Chapter 5 as they analyze diffusion phenomena under the epidemic framework.

Ma and Guo [134] constructed and analyzed four kinds of information transmission patterns within the members of an enterprise under the epidemic transmission framework. In their model, a hyperedge represents an informal organization (e.g., spontaneous groups), and, according to the SIR model, vertices belong to the three standard classes corresponding to ignorant (Susceptible), spreader (Infected), and stifler vertices (Recovered). In the probabilistic transmission model, all vertices are ignorant at the start. At the first iteration, the information spreads from a randomly chosen initial vertex to other randomly selected spreader vertices within the same hyperedge with a given probability. Immune vertices no longer spread the information and stop transmitting. The authors also analyzed variations of the model where information passes (i) from one person to another in a chain-like fashion (one-way transmission), (ii) to the entire hyperedge or group (gossip transmission), or (iii) to a constant number of vertices within the same hyperedge (group transmission).

Suo et al. [173] investigated an SIS model on hypergraphs in the context of rumor spreading on social media. They proposed two information diffusion models by considering how an individual might decide to share content on a social media platform, either to all the contacts or targeting a particular group. In the global strategy, the information is published to the whole network. At each step, an infected vertex i can infect with a probability β all the susceptible neighboring vertices connected to i via a hyperedge. It is worth highlighting that, in this strategy, each hyperedge is seen as a clique. As a consequence, the global spreading strategy would, in principle, be equivalent to the one defined on the two-section of the hypergraph [32] (see Section 2.2.2). With the local approach, an infected vertex i randomly chooses one of its hyperedges e and then tries to infect all the vertices in e with a probability of success β.

5

Analyzing Epidemic Dynamics

Dubium sapientiae initium.

Renè Descartes

In short

This chapter presents current literature related to the analysis of epidemic-spreading phenomena abstracted and studied via hypergraphs. In Section 5.1, we first draw attention to the principal features characterizing equation-based and agent-based models (ABMs), then discuss how these tools are used in the epidemiological context. In Section 5.2, we specifically describe the existing epidemic models on hypergraphs. In Section 5.3, we present an overview of the current literature on modeling non-pharmaceuti- cal interventions via agent-based simulations. In Section 5.4, we finally outline how the parameter space of an ABM can be explored.

5.1 Equation-based vs Agent-based Models

Equation-based models (EBMs) and agent-based models (ABMs) are two modeling approaches for simulating real-world systems and analyzing

what-if scenarios, extensively exploited in pandemic simulations [22]. Both techniques share some common concerns, but they differ in the fundamental relationships among entities they model and the level at which they focus their attention [181].

EBMs and ABMs deal with two entities to model complex problem domains: agents and observables, each with a temporal aspect. [181, 22]. Agents have a specified set of characteristics and interact with each other and their environment according to predefined rules [178]. They may adapt their behavior to their experiences, interactions with other agents, and interactions with their environment. Agents may represent individuals, viruses, governments, or any other entities of interest. Observables are measures of interest that can either be associated with individual agents or with collections of agents. In the context of an epidemic, an example of observable at the agent level is the disease stage the agent is in (such as susceptible, infected, and recovered), while an observable at the collection level is the proportion of the population susceptible. EBMs express relationships among observables: the evaluation of the model produces the observables' evolution over time. EBM tends to make extensive use of system-level observables associated with a collection of interacting agents. On the contrary, ABMs define agent behaviors in terms of observables accessible to the individual agent. Thus, ABMs capture behaviors through which agents interact with one another: the emulation of agent behavior leads to the observables' emergence over time.

EBMs are mathematically tractable and theoretically grounded. However, they assume homogeneity, and they tend to use aggregates of critical system variables. Even though heterogeneity can be incorporated into EBMs, it increases the model's complexity and can make it mathematically intractable. In contrast, ABMs are not mathematically tractable, but they allow for heterogeneity in the problem environment. The price of this feature is that ABMs may require several parameters to capture the agent behavior. Despite creating a much more realistic model, using many parameters also adds computation complexity in terms of sensitivity analysis and model validation. EBMs and ABMs are, thus, two complementary

approaches [22]. While EBMs provide a theoretically justifiable baseline from which ABMs can be calibrated, ABMs can extend EBM's insights by modeling a more realistic decision environment.

5.1.1 Compartmental EBMs and Epidemiology

Modeling a disease transmission introduces a trade-off between a high level of detail and computability. As described by Brauer in [46], there are many mathematical models for studying and analyzing disease spreading. *Compartmental* is a class of models in which the study of transmission and population is divided into compartments, and assumptions on the nature and time rate of the spreading between compartments themselves are made. Popular compartmental models - appropriate for most diseases transmitted by contact - are *SIR* and *SIS*. In these models, the population is studied by partitioning it into three classes labeled S(usceptible), I(nfected), and R(ecoverd). In the SIR model, an individual belongs to one of these possible classes, corresponding to an individual being susceptible to, infected, or recovered from the disease at a particular time. If the disease confers no immunity against infection, individuals cannot move their status to recovered, and they will come back in the susceptible class; in such cases, SIS can be adopted. The pathogen proprieties, such as its contagiousness, the length of its infectious period, and its severity, define the epidemic transmission patterns. However, contagion patterns are also conveyed and defined by the network structure of the population affected by the pathogen [68]. The opportunities for a disease to spread from one individual to another are, thus, given by a contact network. Those are usually modeled via a graph, with members of the population represented by vertices and with contacts between individuals represented by edges. The degree distribution of the so-generated graph becomes fundamental in the description of the spreading process. We refer the reader to the work of Brauer [46] and Nowzari et al. [145] for more details.

5.1.2 ABMs and Epidemiology

An ABM is a class of computational models that provides a bottom-up design approach to define a complex system. As presented by Tracy et al. in [178], ABMs are widely adopted in epidemiology science. Frequently, they are used to integrate GIS to simulate the spread of an epidemic in a particular environment due to individual interactions generated by their mobility over geographical space [109]. ABMs are also suitable for simulating the interactions of autonomous agents, and they can describe a complex system at a micro-scale level [57]. Adopting an ABM in this context can capture several essential aspects of the epidemic dynamics, such as real human behaviors and complex interactions. For instance, we may need to simulate both the direct- and indirect-contagion process between agents and environments to accurately reproduce airborne disease transmission. Further, the duration of the contact is another crucial aspect in these processes as people frequently change their habits according to social events and places' popularity, and distinct pathogens exhibit different infection times. We refer the reader to the work of Tracy et al. [178] for a review of key areas in public health where agent-based modeling has been adopted, including both communicable and non-communicable disease, health behaviors, and social epidemiology.

5.2 Epidemic Models on Hypergraphs

A complete overview of mathematical frameworks capable of explicitly and naturally describing group interactions is given by Battiston et al. [32]. Specifically, the authors outline the dynamics of structures with many-to-many interactions and discuss higher-order diffusion models, including spreading dynamics on hypergraphs. After focusing on the structure of systems with high-order interactions, they describe the mathematical tools that capture many-body relationships, along with the most common measures and properties used to describe systems. The authors further outline the dynamics of structures with many-to-many interactions and discuss models of higher-order diffusion, including spreading dynamics

on hypergraphs.

Bodò et al. [41] first proposed modeling communities as hyperedges, based on the concept that an actual model of an epidemic outbreak has to take into account two factors: community structure and infection pressure. They translated this approach into practice using different contagion probabilities according to the place. In addition, they bounded the likelihood that a susceptible individual becomes infected in a unit to be not proportional to the number of infected individuals within that unit. The authors showed that using a non-linear function to model the infection pressure is crucial not to overestimate the epidemic propagation.

To model the community structure and the non-linear dependence of the infection pressure, Bodò et al. developed the theory of epidemic propagation on hypergraphs, where each node is an individual, and each hyperedge is a unit such as a household or a workplace. Specifically, they examined an SIS model under a continuous time Markov chain formalism. In this model, Poisson processes govern both infection and recovery, where the infection rate r takes into account connectivity patterns. In contrast, recovery is a spontaneous process controlled by a fixed recovery rate γ. In more detail, the probability for a susceptible individual to become infected is defined as $1-e^{-r\Delta t}$, with $r = \tau \sum_e f(i_e)$. The function $f(i_e)$ denotes the number of infected nodes in the hyperedge e, and it is summed up over all hyperedges containing susceptible individuals, while τ represents the infection rate.

To define the non-linear dependence of the infection pressure, Bodò et al. chose f as a linear function as an upper limit to the infection pressure for a susceptible node when the number of infected neighbors is higher than a given threshold. As Battiston et al. note, this approach is conceptually different from the conventional threshold mechanism, in which thresholds set the critical amount of exposure from the peers that an individual needs to adopt new technology. In other words, the function f describes how the infection pressure is discounted as the number of infectious neighbors increases. In particular, if an individual belongs to two or more hyperedges, e.g., a household and a workplace, then the rate of in-

fection is the sum of the rates in each hyperedge, i.e., $(f(n_1) + f(n_2))\tau$, where n_1 is the number of infected individuals at home and n_2 at the workplace. As Bodò et al. observe, the epidemic propagation can also be considered on the corresponding clique graph by discounting the infection pressure for many infectious neighbors. However, in this case, the infection rate will be $f(n_1 + n_2)\tau$ instead of $(f(n_1) + f(n_2))\tau$, and hence it cannot be distinguished whether somebody has more infectious neighbors at home than in the workplace, the vice-versa, or a moderate number at both places.

Simulations on hypergraphs having hyperedges of different sizes indicated that heterogeneous structures might significantly hasten the initial phase of the spreading compared to regular hypergraphs, leading to slightly smaller values of prevalence in the stationary state.

Suo et al. [173] investigated a similar SIS model on hypergraphs in the context of rumor spreading on social media. They proposed two information diffusion models by considering how an individual might decide to share content on a social media platform, either to all the contacts or targeting a particular group. In the global strategy, the information is published to the whole network. At each step, an infected node i can infect with a probability β all the susceptible neighboring nodes connected to i via a hyperedge. It is worth highlighting that, in this strategy, each hyperedge is seen as a clique. As a consequence, the global spreading strategy would, in principle, be equivalent to the one defined on the two-section of the hypergraph [32] (see Section 2.2.2). With the local approach, an infected node i randomly chooses one of its hyperedges e and then tries to infect all the nodes in e with a probability of success β. The two strategies lead to different long-term behaviors, with a vanishing epidemic threshold in the global strategy. In contrast, the particular positioning of the initial seed of infectious nodes — either on high or low degree nodes — seemed to affect only the early evolution of the process: as expected, choosing nodes with a high degree as seeders can significantly speed up the contagion in the early times. No differences in the stationary states were found.

Another version of a high-order contagion model for spreading dynamics occurring at the group level was proposed by Jhun et al. [106]. The authors extended the simplicial contagion model proposed by Iacopini et al. [98] to hypergraphs. Specifically, they studied the spreading process on scale-free d-uniform hypergraphs, where all hyperedges have the same size d. In this model, a susceptible node in a hyperedge e of size d may be infected from e, with rate β_d, only if the remaining $d-1$ nodes composing e are infectious. A standard recovery probability μ is used for recovery. In more detail, the density of infected nodes with degree k, denoted as ρ_k, evolves with time according to $d/d_t\rho_k = -\mu\rho_k + \beta_k(1-\rho_k)k\Theta^{d-1}$. In this formula, the probability that a susceptible node of degree k gets the infection from one of the hyperedges is, as usual, proportional to the infection rate β_k, the number of hyperedges k, and the probability Θ^{d-1} to be connected to a hyperedge having all the other nodes infected.

In their work, Landry et al. [122] focused on studying the effect of degree heterogeneity of the hypergraph structure on epidemic contagion models. They presented and analyzed a hyper degree-based mean-field description of the dynamics of an SIS model, running numerical experiments on hypergraphs where contagion is mediated via both links (pairwise interactions) and triangles (three-way interactions). Under this framework, the authors further considered three group contagion dynamics: (i) collective contagion, (ii) infection by individuals, and (iii) the hipster-effect, mimicking the situation in which a person is less likely to adopt a popular trend, but their pairwise connections can convince them. In the SIS framework Landry et al. proposed, infected nodes heal and become susceptible again at rate γ. The infection process proceeds according to the policy chosen. In the collective contagion case, a susceptible node that belongs to a hyperedge of size m gets infected at rate β_m if all the other members of the hyperedge are infected. This scenario corresponds to the social contagion model based on simplicial complexes of Iacopini et al. [98]. In the individual contagion case, the node gets infected at rate β_m if at least one member is infected. For hyperedges of size 2, both cases reduce to the usual contagion via pairwise interactions.

Recently, de Arruda et al. [61] presented an SIS framework that explicitly includes critical-mass dynamics into the contagion model. The authors generalize the simplicial contagion model proposed by Iacopini et al. [98] both structurally and dynamically. They (i) moved from simplicial complexes to hypergraphs and (ii) allowed a hyperedge e to be potentially infectious for a node $i \in e$ if the number of infected nodes composing e is greater or equal to a given threshold t_e. The authors restricted this threshold mechanism to hyperedges of a size larger than two so that a contagion through active links can always happen (no threshold).

Another very recent study fitting in the framework defined by Bodò et al. has been proposed by Higham et al. [92]. The authors inherited the idea of a nonlinear infection pressure from each hyperedge (as well as the notion of hyperedge as a place, like a household, workplace, or social setting) and studied the case where the form of the nonlinearity depends on the hyperedge type. The authors presented both the exact individual-level stochastic model and a deterministic mean-field approximation based on an SIS high-order propagation model. They further discussed different types of contagion models in this hypergraph setting and derived spectral conditions characterizing whether the disease vanishes. Higham et al. also showed how the hypergraph model allows distinguishing between contributions to infectiousness due to (i) the nature of the pathogen - modeled via the infection parameter β - and (ii) behavioral choices (such as social distancing, increased hygiene, and use of masks) - represented by the interaction structure of the hypergraph and a coefficient arising from modeling the nonlinear infection process.

Finally, Ma and Guo [134] constructed and analyzed four kinds of information transmission patterns within the members of an enterprise. In their model, a hyperedge represents an informal organization (e.g., spontaneous groups), and, according to the SIR model, nodes belong to the three standard classes corresponding to ignorant (S), spreader (I), and stifler nodes (R). In the probabilistic transmission model, all nodes are ignorant at the start. At the first iteration, the information spreads from a

randomly chosen initial node to other randomly selected spreader nodes within the same hyperedge with a given probability. Immune nodes no longer spread the information and stop transmitting. The authors also analyzed variations of the model where information passes (*i*) from one person to another in a chain-like fashion (one-way transmission), (*ii*) to the entire hyperedge or group (gossip transmission), or (*iii*) to a constant number of nodes within the same hyperedge (group transmission).

5.3 Modeling Non-Pharmaceutical Interventions

Non-pharmaceutical interventions (NPIs) have been the subject of vast literature even before the COVID-19 pandemic [77, 182]. Still, until 2007, Aledort et al. report a generally poor quality of evidence on which to base non-pharmaceutical pandemic planning decisions [6], mainly due to the lack of representative data and a validation process [182]. Unfortunately, when the COVID-19 started spreading worldwide, NPIs were the only possible measures to stop its diffusion. This event gave birth to an unseen joint effort of the academic community and tech giants in understanding, modeling, and assessing the connection between human behaviors and disease diffusion and the effects of the application of NPIs [153]. However, despite the current availability of vaccines, from a study performed on the UK region [137], the application of such measures cannot be completely relaxed as the current vaccination program alone is insufficient to contain the outbreak.

In a recent survey [153], Nicola Perra thoroughly describes current literature about NPIs during the COVID-19 pandemic. The author classifies the models adopted into four categories: (*i*) compartmental models, (*ii*) metapopulation models, (*iii*) statistical models, and (*iv*) agent-based models. In this section, we will focus on agent-based models as, in this dissertation, we exploit the ABM paradigm to simulate the epidemic spreading and the application of NPIs. As described in Section 5.1, an ABM is a class of computational models that provides a bottom-up design approach to define a complex system, offering an easy way to plug socio-

demographic features into the simulation of a disease spreading. Nevertheless, the price to pay for having a realistic model lies in the specification of several parameters to capture the agent behavior and the computation complexity in terms of sensitivity analysis and model validation [22]. ABMs have been widely used to quantify the effects of NPIs, such as lockdown, social distancing, and isolation measures. Another recent survey about the use of ABMs to simulate the COVID-19 pandemic can be found in [130].

Hoertel et al. [93] developed a detailed ABM for France to evaluate the effectiveness of different NPIs in the reopening phases after the first wave. The proposed model has 194 parameters, describing the socio-demographic features of the French population (140), the contact networks (33), and the features of the virus (21). The authors studied lockdown and post-lockdown measures, including physical distancing, mask-wearing, and isolation approaches, highlighting how the interventions after lifting the first lockdown were not enough to overburden the healthcare system. With a similar approach, Aleta et al. [8] built a multilayer synthetic population that models the socio-demographic features of the Boston metropolitan area using high-resolution data describing the movements and potential interactions of people in the city to investigate the impact of different reopening scenarios. Their results suggest how a proactive policy of testing, contact tracing, and household quarantine could gradually reopen economic activities and workplaces with a low impact on the healthcare system. Analogous to this work, the same authors also proposed an ABM for the metropolitan areas of New York and Seattle [7], informing the model with mobile phone data and Foursquare data to identify POIs. In this study, the authors examine the risk of infection according to the type of locations visited and estimate the size of transmission chains, finding that workplaces, restaurants, and grocery stores are the main drivers of the epidemic patterns.

Wilder et al. [188] proposed an ABM to study the spreading of COVID-19 in Hubei, Lombardy (Italy), and New York City. The synthetic population is formed by individuals stratified for age, comorbidities, and as-

signed to a household, while contacts among agents in different contexts than home are modeled via contact matrices. Their results suggest that measures should be tailored to the specific socio-demographic features of each population as the efficacy of NPIs varied across the analyzed location. Contact matrices were also used by Ogden et al. [146] to model a synthetic sample of the Canadian population to evaluate social distancing and isolation measures to control the disease spread. Their outcomes indicate that lifting disruptive NPIs such as shut-downs must be accompanied by enhancements to other NPIs to prevent new introductions and identify and control new transmission chains.

Yang et al. [191] used a network-based model to represent contact happening inter- and intra- different cities in the Hubei province. They specifically examined the use of personal protective, social distancing, and a combination of those measures to decrease the infection rate.

Bouchnita et al. [45] designed a multi-scale ABM in which agents move according to a social force model and which considers both direct and indirect transmission mechanisms. The authors did not explicitly consider the notion of location (as households or schools) as agents move on a grid. The model simulates indirect contagion based on the normalized concentration of deposited SARS-CoV-2 on hard surfaces, the averaged rate of SARS-CoV-2 secretion by contagious agents, and the decay rate of the virus. For each agent, indirect transmission can occur only once every day at a random moment.

Silva et al. [167] proposed an agent-based framework (COVID-ABS) to simulate people, business, government, and the health care system. The framework allows implementing several NPIs and measures the impact in terms of disease and economic burden. Other few works also use ABMs to simulate the economic consequences of the COVID-19. For instance, Inoue and Todo [99] quantified the economic effect of a possible lockdown of Tokyo, estimating that the lockdown would result in an 86% reduction of the daily production in Japan after one month. Similar to COVID-ABS, Dignum et al. [64] presented an ABM simulation tool to analyze possible repercussions of policy interventions, combining social,

economic and health aspects.

Generally, all works agree that continued intervention should be considered to keep the transmission of an epidemic under control and mix NPIs to regulate contagion dynamics best.

5.4 Model Exploration

ABMs are usually characterized by a large number of controlling parameters and range of parameter values, a significant amount of computation required to run a model, and a stochastic nature which requires multiple trials to assess the model's behavior [171]. As the number of parameters increases, it becomes unfeasible to manually handle the exploration of the parameter space, which comprises the selection of parameter selection, the simulation run, and the evaluation of the output [50]. The process of (i) choosing starting configurations, (ii) running the simulation, (iii) evaluating the outcomes, and (iv) selecting new candidates is known as Simulation Optimization (SO) process [76, 177]. A SO framework can be formally described as a general optimization problem whose goal is to find a setting of controllable parameters that minimizes a given objective function, i.e.,

$$\min_{\theta \in \Theta} J(\theta),$$

where Θ is the admissible decision space, $\theta \in \Theta$ is the vector of input variables representing a single configuration, and $J(\theta)$ is the scalar objective function estimated via the simulation. Because of their nature, simulations provide a noisy estimate of $J(\theta)$; for this reason, the most common form for J is an expectation

$$J(\theta) = \mathbb{E}[L(\theta, \epsilon)],$$

where $L(\cdot)$ is the sample performance measure and ϵ represents the stochastic effect in the system.

Having formalized the SO process as a general optimization problem, the parameter space search can thus be done via any optimization algo-

rithm [177]. The literature widely explored genetic algorithms (GAs) in this direction, working on parameter-search and exploration in ABMs, as well as on the problem of parameter-search in general [171]. When the optimization problem is multi-objective ($J(\theta) \in \mathcal{R}^n$), multi-objective GAs (MOGAs) come into play. Their central idea lies in generating the optimal Pareto frontier in the objective space so that it is not possible to further enhance any fitness function without disturbing the other fitness functions [127]. The goals of MOGAs are convergence, diversity, and coverage.

These algorithms can be broadly categorized in two classes: (i) Pareto-based MOGAs, and (ii) decomposition-based MOGAs. The algorithm NSGA [169], its enhanced version (NSGA-II [62]), and variants (NSGA-II with dynamic crowding distance [133]) are among the main examples of Pareto-based MOGAs. Decomposition-based MOGAs decompose the given problem into multiple subproblems - which are solved simultaneously - and exchange the solutions among neighboring subproblems. Examples of this class of MOGAs are: MOGLS [100] and its variant [101], C-MOGA [141], MOTGA [9], and D-MOGA [151].

For an exhaustive description of the SO process and a review on genetic algorithms, we refer the reader to [76, 177] and [127, 115], respectively.

6

Data Sets

> Without data you're just another person with an opinion.
>
> Edwards Deming

In short

In this chapter, we introduce the data sets used for the experimental part of this dissertation. We provide a brief description for each data set, a link to the source where it is possible to download the data, and some basic information about the number of instances and the available features. Whenever more detailed analyses are needed, we report them in the specific chapter.

6.1 Data Sets Description

In the following, we list in alphabetical order all the data sets used in this thesis.

Algebra

Hypergraph where vertices are users of `mathoverflow.net` and hyperedges correspond to users who answered a particular type of question about algebra within a month. The network has 423 vertices and $1,268$ hyperedges. Source [35].

Used in Chapter 8.

Amazon

Sets of products reviewed by users on Amazon. Each vertex corresponds to a product, and a hyperedge links together groups of similar items. The network has $2,268,231$ vertices and $4,285,363$ hyperedges. In our experiments, we used a subset of reviews. We refer the reader to Section 8.5.2 for details on the hypergraph size. Source [35].

Used in Chapter 8.

Bars Reviews

Hypergraph where vertices are Yelp users, and hyperedges are users who reviewed an establishment of a particular category (different types of bars in Las Vegas, NV) within a month time frame. The network has $1,234$ vertices and $1,194$ hyperedges. Source [35].

Used in Chapter 8.

BLEBeacon

The BLEBeacon data set [166] is a collection of Bluetooth Low Energy (BLE) advertisement packets/traces generated from BLE beacons carried by people following their daily routine inside a university building for a whole month. The main objective of this data set was to present a real-life realization of a location-aware sensing infrastructure that can provide insights for smart sensing platforms, building management, and user-localization frameworks. The data set contains $153,868$ check-ins of 62 users and 31 locations.

Used in Chapter 10.

Blues Music Reviews

Hypergraph where vertices are Amazon reviewers and hyperedges are reviewers who reviewed a particular product category (different types of blues music) within a month timeframe. The network has $1,106$ vertices and 694 hyperedges. Source [35].

Used in Chapter 8.

DBLP

Co-authorship on DBLP papers, containing 2 million documents scraped from DBLP. Each vertex is an author, while each hyperedge is a publication. In our experiments, we considered all documents published between January and May 2017. We refer the reader to Section 8.5.2 for details on the hypergraph size. Source [56].

Used in Chapter 8.

Email-Enron

Sets of email addresses on emails. In this dataset, vertices are email addresses at Enron, and a hyperedge is comprised of the sender and all recipients of the email. The network has $4,423$ vertices and $15,653$ hyperedges. In our experiments, we considered a subset of emails. We refer the reader to Section 8.5.2 for details on the final hypergraph size. Source [35].

Used in Chapter 8.

Email-W3C

Sets of email addresses on emails. Each hyperedge consists of a set of email addresses, which have all appeared on the same email. The network has $14,317$ vertices and $19,821$ hyperedges. In our experiments, we considered a subset of emails. We refer the reader to Section 8.5.2 for details on the final hypergraph size. Source [35].

Used in Chapter 8.

Foursquare

The Foursquare social network data set, introduced by Yang et al. in [190], is a collection of check-ins originated from the city of Tokyo and crawled from 12 April 2012 to 16 February 2013. The data set contains $573,703$

check-ins of $2,293$ users and $61,858$ locations, such as restaurants, cinemas, sports, and so on.

Used in Chapters 9 and 10.

Game of Thrones (GoT)

Data set on the GoT TV series describing season episodes and containing meta-informa- tion about each of them, such as title, identification number, season, and description. Information about each scene within an episode is also reported. For each scene, start, end, location, and a list of characters performing in it are listed. A more detailed description of the data set can be found on the GitHub repository of the creator [103]. Table 6.1 reports some basic information about the number of episodes, scenes, and characters per GoT season.

In Section 7.4.2 and Chapter 8, we modeled the GoT data set with a hypergraph, whose vertices are GoT characters and hyperedges are GoT scenes. In other words, a hyperedge consists of all characters appearing in the same scene together.

Used in Chapters 7 and 8.

Table 6.1: Number of episodes, scenes, and characters per GoT season.

Season	Episodes	Scenes	Characters
I	10	286	125
II	10	468	137
III	10	470	137
IV	10	517	152
V	10	508	175
VI	10	577	208
VII	7	468	75
VIII	6	871	66

Geometry

Hypergraph where vertices are users on `mathoverflow.net`, and hyperedges are sets of users who answered a certain question category about geometry. The network has 580 vertices and $1,193$ hyperedges. Source [35].

Used in Chapter 8.

NBA

NBA games in the period 1985-2013. This data set has been collected by Sports Reference LLC and contains around 32K nested documents representing NBA games. Each document represents a game between two teams with at least 11 players each. It contains 47 attributes; 40 of them are numeric and represent team and player results. Vertices are players, and a hyperedge connects together all players involved in a match up to 2012. We refer the reader to Section 8.5.2 for details on the final hypergraph size. Source [56].

Used in Chapter 8.

Restaurant Reviews

Hypergraph where vertices are Yelp users, and hyperedges are users who reviewed an establishment of a particular category (different types of restaurants in Madison, WI) within a month timeframe. The network has 565 vertices and 601 hyperedges. Source [35].

Used in Chapter 8.

Yelp.com

Yelp.com [192] is an online platform where customers can share their experiences about local businesses by posting reviews, tips, photos, and videos. Further, this platform allows businesses and customers to engage and transact. Every year, the Yelp Inc. Company releases part of their data as an open data set to grant the scientific community to conduct research and analysis on them. In this dissertation, we analyzed the 2019

Yelp Challenge data set [193], containing information about businesses, reviews, and users. Table 6.2 describes all the accessible data set entities. A more detailed description can be found on the official page [194]. We have to note that the Yelp data set is regularly updated; thus, the current snapshot may not be precisely the same as the one in 2019.

Used in Chapter 7.

Table 6.2: Yelp entities contained in the data set.

Data	Instances	Description
Business	192,609	Business data including location, attributes, and categories.
User	1,637,138	User data including the user's friend mapping and all the metadata associated with the user.
Review	6,685,900	Full review text including the `user_id` that wrote the review and the `business_id` the review is written for.
Picture	200,000	Photo data including caption and classification (one of "food", "drink", "menu", "inside" or "outside").
Tip	1,223,094	Tips written by users on businesses. Tips are shorter than reviews and tend to convey quick suggestions.
Check-in	192,609	Aggregated check-ins over time for each business.

PART

II

Tools for Hypergraphs

Summary

7

SimpleHypergraphs.jl

Simplicity is the soul of efficiency.

Austin Freeman

In short

This chapter presents one of the main contributions of this thesis: SimpleHypergraphs.jl, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing. Here, we describe the main motivations behind creating SimpleHypergraphs.jl (see Section 7.1), the library's design choices (see Section 7.2), and its memory model. We further delve into illustrating the functionalities offered by the software, including transformations to graphs and hypergraph visualization methods (see Section 7.3). Contextually, we also describe a generalized version of the label propagation algorithm for community detection suitable for hypergraphs. We finally present two case studies with the twofold objective of (*i*) demonstrating how it is possible to exploit the proposed library and (*ii*) comparing hypergraphs with their corresponding graph counterpart to explore whether high-order structures convey more information in addressing specific tasks (see Section 7.4).

SimpleHypergraphs.jl is available on a public open-source GitHub repository [12] and on the official Julia package registry.

The work described in this chapter has been presented in the following articles:

- A. Antelmi, C. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, P. Szufel. SimpleHypergraphs.jl — Novel Software Framework for Modelling and Analysis of Hypergraphs. In: *Algorithms and Models for the Web Graph*, pages 115-129, Cham, 2019. Springer International Publishing.

- A. Antelmi, C. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, P. Szufel. Analyzing, Exploring, and Visualizing Complex Networks via Hypergraphs using SimpleHypergraphs.jl. *Internet Mathematics*, 2020.

7.1 Motivation

Working with hypergraphs requires software libraries specifically designed to perform operations directly on these high-order structures, from basic algorithms (e.g., traversals) to more complex tasks (e.g., community detection). This desideratum represents the fundamental requirement for each hypergraph-specific framework, in addition to a flexible definition of the data structures and functionalities exposed to allow the user to customize their behaviors easily if needed. Further, an extra point goes to the implementation language, which should be at the same time easy-to-learn and fast to enable the implementation of efficient algorithms.

In Section 3.1, we discussed the major existing libraries to represent and manipulate hypergraphs. However, from the analysis of the functionalities offered by each software emerge that the available frameworks are a compromise between efficiency, characterizing low-level languages such as C/C++, and the ease-of-use and expressiveness, peculiar of scripting languages such as Python and R. In particular, HyperNetX [155], which currently stands out as a comprehensive framework written in Python for

analyzing hypergraphs, was only in its early development stage in 2019. Further, some libraries only support a very restricted set of hypergraph-related functions or only rely on hypergraph to graph transformations and do not expose any specific method.

To address the specific needs listed in the introduction of this section and given the lack of both a comprehensive and efficient hypergraph-specific library, we opted to develop our software framework for the Julia language. We chose this programming language thanks to its peculiar characteristics of being a high-level language designed explicitly for high-performance computation (see Section 3.2), hence perfectly suitable for our task. Furthermore, no support for hypergraphs existed yet in the Julia community. Today, SimpleHypergraphs.jl is the reference library to model, manipulate, and visualize hypergraphs in Julia.

Table 7.1 summarizes the main characteristics of SimpleHypergraphs.jl and the principal libraries introduced in Section 3.1. We left out from the comparison currently unmaintained libraries, newly and project-like frameworks, and software without hyper- graph-specific functionalities. The *Graph Integration* and *Visualization* entries refer to the integration of a graph-specific library within the hypergraph framework and the support for hypergraph visualization, respectively.

7.2 Library Design

The key element in the design of SimpleHypergraphs.jl is to ensure flexible interoperability with both the existing Julia ecosystem and the classical graph manipulation framework. For this reason, SimpleHypergraphs.jl provides two-fold integration with both the Julia standard matrix type and the LightGraphs.jl package, the official Julia library to manipulate graphs.

Julia's matrix APIs. The integration with Julia's built-in type `Array` is obtained by making the `Hypergraph` struct a subclass of `AbstractMatrix`, and providing a set of integration methods for manipulating matrices, such as querying the matrix size and fetching/updating ele-

Table 7.1: Summary of the hypergraph software frameworks' main characteristics. *CommDet* stands for Community Detection.

	SimpleHypergraphs.jl	CHGL	halp	Hygra	HyperNetX
Reference	[12]	[105]	[139]	[163]	[155]
Language	Julia	Chapel	Python	C/C++	Python
License	MIT	MIT	GPL-3.0	MIT	3-Clause BSD
Documentation	✓	Work in Progress	✓	✓	✓
Generative Models	✓	✓	✓	×	✓
Metadata	✓	×	×	×	✓
Parallel / Distributed	×	✓	×	✓	✓
Hypergraph algorithms					
- Centrality Measures	✓	×	×	✓	✓
- Random walk/Paths	✓	✓	✓	✓	✓
- CommDet/Clustering	✓	×	✓	×	✓
Graph integration	✓	×	✓	✓	✓
Visualization	✓	×	×	×	✓

ments (see §Querying and Manipulating Functions in Section 7.3.2).

Internally, a hypergraph is stored as a sparse array, and its data are stored in a redundant format, using two separate hashmap structures for rows and columns. This design choice simultaneously guarantees good algorithmic performances on rows and columns. Furthermore, it avoids the circumstance where all data need to be rewritten when the adjacency matrix is updated (typical disadvantage of a compressed sparse row matrix).

As a subclass of AbstractMatrix, the hypergraph adjacency matrix can be manipulated like any other Julia matrix (see Section 7.3.1).

LightGraphs.jl. We achieved interoperability with the LightGraphs.jl library by creating hypergraph "view" classes that act as an interface and represent a hypergraph as either an incidence or a clique graph (see §Hypergraph Transformations in Section 7.3.2). These representations do not copy the hypergraph data but provide a view that

allows access to the underlying hypergraph in a read-only mode. As we developed a comprehensive set of integration methods for the LightGraphs.jl package, the user can directly analyze a hypergraph structure (transformed into a graph) with all functionality provided by LightGraphs.jl.

In the following, we describe all the functionalities that SimpleHypergraphs.jl supports at the time of writing.

Graphs.jl

As of 8 October 2021, LightGraphs.jl is no longer under active development, and all its functionalities have been merged within the new Graphs.jl Julia package. The older versions of LightGraphs can be used indefinitely; however, we will shortly update this dependency.

7.3 Library's Internals and Supported Functions

The latest version v0.1.15 of SimpleHypergraphs.jl, released on the 9th June 2021, provides a wide range of functionalities to build and explore hypergraphs. The following sections introduce the hypergraph representation, several basic querying and manipulating operations, and graph transformations. We further describe two serialization mechanisms, a set of analytical algorithms, and two visualization methods.

7.3.1 Memory Model

SimpleHypergraphs.jl provides APIs representing a hypergraph $H = (\mathcal{V}, E)$ as a matrix $\mathbf{H} \in \mathbb{R}^{n \times m}$, where n is the number of vertices and m is the number of hyperedges. Thus, each row of the matrix $\mathbf{H}$ is associated with a vertex and indicates the hyperedges the vertex belongs to (see §Matrix Notation in Section 2.1). In the APIs, vertices and hyperedges

are uniquely identified by progressive integer ids, corresponding to rows $(1, \ldots, n)$ and columns $(1, \ldots, m)$, respectively. Each entry $\mathbf{H}(i, j)$ of the matrix denotes the weight $\lambda_j(i)$ of the vertex i within the hyperedge j. The library provides several constructors and enables attaching metadata values of arbitrary type to both vertices and hyperedges.

Hypergraph Constructors. The Julia hypergraph object is defined as:

```
Hypergraph{T, V, E, D} <: AbstractMatrix{Union{T, Nothing}}
```

where `T`, a subtype of `Real`, represents the type of the weights stored in the structure; `V` and `E` are the types of the metadata values stored in the vertices and hyperedges of the hypergraph, respectively; and `D`, a subtype of `AbstractDict{Int,T}`, is the type of the underlying dictionary used for storing the weight values.

When calling the constructor, the parameters `{T, V, E, D}` can be omitted, starting from the rightmost. The default value for the dictionary type `D` is a standard Julia dictionary `Dict{Int, T}` (where `T` is the type of the weights). However, the user can adopt a different dictionary implementation (e.g., a sorted dictionary) to ensure the reproducibility of the results (for instance, when running stochastic simulations relying on the hypergraph structure). The default value for vertex and hyperedge metadata types, `V` and `E`, is `Nothing` - i.e., by default, no metadata is stored.

A new empty hypergraph can be built specifying the number of vertices (rows) and hyperedges (columns). Optionally, a hypergraph can be either materialized starting from a given matrix or a LightGraphs.jl graph object.

7.3.2 Functionalities

A description of the main functionalities supported by SimpleHypergraphs.jl follows. Each method is accompanied by an extensive documentation.

Generative Models

SimpleHypergraphs.jl supports the automatic generation of random hypergraphs either with or without any specific structural constraints. A more detailed description of the four generative models implemented is reported below.

Random model. This method generates a hypergraph without any structural property constraint. Given two integer parameters n and m (the number of vertices and hyperedges, respectively), the algorithm computes - for each hyperedge $e \in [1, m]$ - a random number $s \in [1, n]$ (i.e., the hyperedge size). Then, the algorithm selects uniformly at random s vertices from V to add in e.

$k-$*uniform model*. This method generates a k-uniform hypergraph (see Section 2.1). The algorithm proceeds as for the random model but forcing the size of each hyperedge to be equal to k.

$d-$*regular model*. This method generates a d-regular hypergraph (see Section 2.1). The algorithm exploits the k-uniform approach described above to build a k-uniform hypergraph H having m vertices and n edges. It then returns the dual hypergraph H^*.

Preferential-attachment model. This method generates a hypergraph with a preferential attachment rule between vertices, as described in [25]. The algorithm starts with an entirely random graph with five vertices and five hyperedges. It then iteratively adds a vertex or an edge, according to a given parameter p, defining the probability of creating a new vertex or a new hyperedge. Specifically, the connections with the new vertex or hyperedge are generated according to a preferential attachment policy [25]. We slightly modified the algorithm to avoid repetitions in the hyperedges.

Querying and Manipulating Functions

SimpleHypergraphs.jl provides the following accessing and manipulating functions. We only list hypergraph-generic and vertex-specific methods, but analogous functionalities are also implemented for the hyperedges.

- `size` returns the size of the hypergraph H as a tuple (number of vertices, number of hyperedges);
- `getindex` returns a value for a given vertex-hyperedge pair for a hypergraph H;
- `setindex` removes or adds a vertex from a given hyperedge in a hypergraph H and a given vertex-hyperedge pair;
- `get_vertices` returns the vertices for a given hyperedge in H;
- `add_vertex!` adds a vertex to a given hypergraph H. Optionally, the vertex can be added to existing hyperedges. Additionally, a value can be stored with the vertex using the `vertex_meta` keyword parameter;
- `remove_vertex!` removes a vertex from a given hypergraph H;
- `set_vertex_meta!` sets a new meta-value for a vertex;
- `get_vertex_meta` returns the meta-value stored in a vertex;
- `nhv` returns the number of vertices in the hypergraph H;
- `prune_hypergraph` removes all $0-$degree vertices and $0-$size hyperedges.

Hypergraph Transformations

The library allows the user to transform a given hypergraph into its corresponding two-section or incidence graph. We have introduced both representations in Section 2.2.2, but we recall their definition in the following.

`TwoSectionView`. This type is the two-section representation of a hypergraph. The two-section graph $[H]_2$ of a hypergraph H is a graph whose vertices are the vertices of H and where two distinct vertices form an edge if and only if they are in the same hyperedge. As a result, each hyperedge from H occurs as a complete sub-graph in $[H]_2$. It is also possible to build the weighted two-section $[H]_2^w$ of H, in which the weight of an edge corresponds to the number of hyperedges that contain both the endpoints of the edge.

`BipartiteView`. This type is the incidence graph of a hypergraph H. The incidence graph of H is the bipartite graph $I(H) = (\mathcal{V}', \mathcal{E}')$ with a vertex set $\mathcal{V}' = \mathcal{V} \cup E$, and where $v \in \mathcal{V}$ and $e \in E$ are adjacent if and only if $v \in e$.

For performance reasons, both representations are views of the actual hypergraph, hence changes to the original hypergraph structure will be automatically reflected in the corresponding graph transformation. Further, both `Views` are instances of `AbstractGraph`, the graph object defined by the LightGraphs.jl library. This implies that all methods exported by LightGraphs.jl are callable on both representations. When the view is materialized, the generated graph does not include any meta information.

Julia composite types

Composite types are called records, structs, or objects in various languages. In object oriented languages, such as Java and Python, composite types also have named functions associated with them,

> and the combination actually form an *object*. In Julia, all values are objects, but functions are not bundled with the objects they operate on. This is necessary since Julia chooses which method of a function to use by multiple dispatch, meaning that the types of all of a function's arguments are considered when selecting a method, rather than just the first one (see Section 3.2).

Hypergraph Serialization

The library currently offers two mechanisms to load and save a hypergraph from or to a stream. Given a hypergraph H, it may be stored using either a *plain text* or *JSON* formats. Both formats are human-readable.

Plain text format, denoted by the `HGF_Format` storage type. The first line consists of two integers n and m, representing the number of vertices and the number of hyperedges of H, respectively. The following m rows (lines within the text file) describe the actual structure of H: each line represents a hyperedge and contains a list of all vertex-weight pairs within that hyperedge.

JSON format, denoted by the `JSON_Format` storage type. The internal hypergraph structure is represented with a dictionary that is serialized into a plain JSON object. Each dictionary key represents a hypergraph field, while each dictionary value stores the corresponding hypergraph field value. Additionally, the matrix view of H is also stored. We used the Julia package `JSON3.jl` to handle the interaction between JSON and Julia types.

Analytical Functionalities

Centrality Measures. SimpleHypergraphs.jl provides support for the evaluation of a vertex (hyperedge) degree, betweenness, and closeness

centrality measures, based on the notion of s-paths (see §Centrality Measures in Section 2.1). Specifically, each centrality measure can be evaluated across all vertices of the input hypergraph or on a specified subset of vertices. A vector representing the centrality calculated is returned.

Community Detection. Discovering communities in complex systems is a primary approach to understanding how the network structure relates to the underlying system behaviors, thus helping to find out hidden interaction patterns [96]. The term community generally refers to a group of highly connected vertices but loosely linked to other vertices in the network. SimpleHypergraphs.jl offers two different methods to find communities within a hypergraph, one based on the notion of modularity and one generalizing the label propagation algorithm proposed for graphs.

Community detection via hypergraph modularity. The notion of modularity is a key ingredient when studying a network's structure. It is simultaneously a global criterion to define communities, a quality function of community detection algorithms, and a way to measure the presence of community structure in a network. Newman and Girvan [142] first introduced the concept of modularity for graphs, based on the comparison between the actual density of edges inside a community and the density one would expect to have if the vertices of the graph were attached at random regardless of community structure while respecting the vertices' degree on average. Higher modularity values signify denser connections between the vertices within clusters but more sparse connections between vertices in different clusters.

In SimpleHypergraphs.jl, we implemented the notion of hypergraph modularity proposed by Kamiński et al. in [113]. In addition, we also provided an algorithm to calculate the modularity of a given vertex partition. This functionality is achieved via the `modularity` function, which takes a hypergraph and its partition as inputs.

Community detection via label propagation (*LP*). The LP algorithm for graphs was proposed by Raghavan et al. [157] in 2007. The high-level algorithm strategy can be summarized as follows. Initially, each vertex has a unique label (initialization phase). Then, every vertex updates its label at each iteration according to the most frequent among its neighbors (propagation rule). If multiple choices are possible, the new label is randomly picked among those. The algorithm ends when there are no more changes in the vertices' labels or after a predefined number of iterations (termination criteria).

In SimpleHypergraphs.jl, we implemented a generalized version of the LP algorithm just described, designed ad-hoc for hypergraphs. The proposed algorithm shares the initialization phase and the termination criteria with the standard label propagation algorithm proposed for graphs. The critical difference lies in the propagation rule, which, in this case, is composed of two sub-phases: a hyperedge labeling phase and a vertex labeling phase. During the hyperedge labeling phase, the labels of the hyperedges are updated according to the most frequent label among the vertices contained in the hyperedge. Similarly, during the vertex labeling phase, the label of each vertex is updated by choosing the label that is the most frequent among the hyperedges it belongs to.

Both algorithms are callable via the function `findcommunities`. The correct algorithm to run is chosen thanks to the multiple dispatch functionality of Julia, as the user needs only to specify the *type* of the algorithm.

Connected Components. SimpleHypergraphs.jl supports the evaluation of a hypergraph's connected components. Specifically, the function `get_connected_components` takes a hypergraph H as input and returns a vector of vectors, where each element represents a set of vertices that are within the same connected component in H.

Random Walks. Random walks on graphs have been extensively used for a variety of graph-based problems such as ranking vertices, predicting links, recommendations, and clustering. However, while a walk on a graph is a sequence of vertices, a walk in a hypergraph is a vertex-hyperedge alternating sequence (see Section 2.1). Hence, each random walk scheme in a hypergraph should account for the double-step happening in each hyper-path: the passage vertex-to-hyperedge and hyperedge-to-vertex.

One of the most generic random walk schemes can be the following approach. Let the walk start from some vertex v. Then, the vertex v can randomly select a hyperedge e it belongs to, and next choose a target vertex within e, again at random. SimpleHypergraphs.jl provides the function `random_walk`, which runs one step of the walk: it takes a hypergraph and a starting vertex as inputs and returns a destination vertex. This function also accepts two optional functions as keyword arguments, `heselect`, and `vselect`. The former specifies the rule by which a hyperedge is selected for a given starting vertex. The latter selects the destination vertex from the selected hyperedge. By default, `heselect` chooses a hyperedge containing the source vertex uniformly at random. Similarly, `vselect` selects a vertex from a given hyperedge uniformly at random. This design choice guarantees complete flexibility in defining random walks on hypergraphs, allowing the user to define their selection functions.

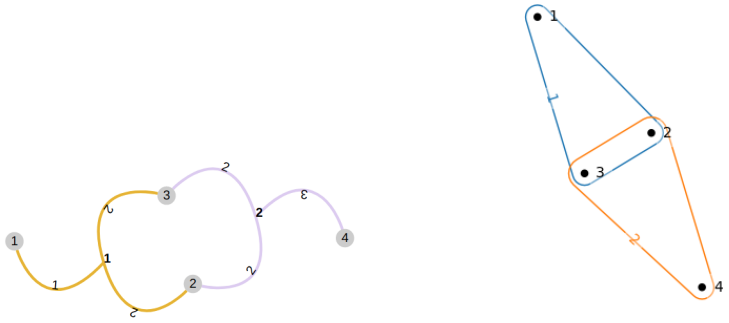
SimpleHypergraphs.jl also provides the implementation of a high-order random walk for hypergraphs proposed by Bellaachia and Al-Dhelaan in [34].

Hypergraph Visualization

SimpleHypergraphs.jl provides the possibility to draw a hypergraph by exploiting two kinds of visualization, both accessible via the function `draw`. The available plotting methods either exploit an interactive JavaScript (JS) or a static Python-based solution. Figure 7.1 illustrates the same hypergraph drawn using the two different strategies. A more detailed description follows.

JS-based visualization. When dealing with complex objects that need to be visualized, it is of fundamental importance to have the possibility to easily catch the main information and, at the same time, to be able to retrieve more detail on demand [179]. For this reason, we decided to integrate a dynamic and interactive visualization within SimpleHypergraphs.jl. This visualization is a wrapper around an external JS package that exploits D3.js, a JS library for manipulating documents based on data, which combines powerful visualization components and a data-driven approach to DOM manipulation. This architectural stack provides the user with a way to generate a dynamic visualization embeddable into a web-based environment, such as a Jupyter Notebook. This method represents each hyperedge e of an hypergraph H as a new fictitious vertex fv to which each vertex $v \in e$ is connected (see Figure 7.1a). The appearance of vertices and hyperedges, whether displaying vertex weights and vertex and hyperedge metadata and labels are customizable.

Python-based visualization. This visualization is a wrapper around the drawing functionalities offered by the Python library HyperNetX [155], built upon the Python package matplotlib. HyperNetX renders an Euler diagram of the hypergraph where vertices are black dots and hyperedges are convex shapes containing the vertices belonging to the edge set (see Figure 7.1b). As the authors note, it is not always possible to render the *correct* Euler diagram for an arbitrary hypergraph. For this reason, this technique may lead to cases where a hyperedge incorrectly contains a vertex not belonging to its set. This library allows the user to manipulate the appearance of the resulting plot by letting the user defining the desired label, vertex, and edge options. SimpleHypergraphs.jl fully integrates the visualization potentiality of HyperNetX.



(a) JavaScript visualization. Each hyperedge is transformed into a fictitious vertex to which each vertex is connected.

(b) HyperNetX visualization. The hypergraph is visualized as an Euler diagram.

Figure 7.1: SimpleHypergraphs.jl visualization methods.

7.4 Use Cases

In this section, we present two practical applications of SimpleHypergraphs.jl to analyze as many real-world hypernetworks. Specifically, the first case study deals with customers reviewing businesses on the social platform *Yelp.com*, while the second application investigates the relationships between the Game of Thrones (GoT) TV Series characters. In each experiment, we compare the hypergraph and the graph-based approaches in tackling the same problem to explore whether high-order analytical tools actually convey more information when many-to-many interactions intrinsically characterize the underlying structure of the data.

7.4.1 Exploring and Analyzing User Reviews: the Yelp.com Case

The heterogeneity and massive quantity of data released by the online platform Yelp.com allow the modeling of a myriad of complex interactions (see Section 6.1). In this use case, we specifically focus on represent-

ing the relation *user-review-business* with a hypergraph, in which a vertex represents a business, while a hyperedge symbolizes a user that has reviewed that particular business.

As for the GoT use case, the main research question underpinning our experiments is the following:

> Does modeling the Yelp.com data set with hypergraphs give qualitatively more information than looking at the corresponding two-section graph representation?

To answer such a question, we set up two different experiments. In the first experiment, our aim is to forecast the number of stars of a business v , given the information available in the neighborhood of v. In the second experiment, we focus on understanding review patterns. We have to note that the principal aim here is not to solve the specific task in a fancy or complicated way. Rather, we preferred implementing more straightforward approaches to obtain easy-to-interpret results. In the following, we describe the data sets used for these analyses, both experiments, and the insights obtained.

Modeling Yelp.com via Hypergraphs

We modeled the Yelp.com data set using a hypergraph $H = (\mathcal{V}, E)$, where the set of vertices $\mathcal{V}$ represents Yelp businesses, and the family of hyperedges E symbolizes Yelp users. Specifically, each hyperedge denoting a user u contains all businesses u has written at least one review for. Given the massive amount of available data (around 9 GB), we only explored a subset of the whole Yelp.com data to build two different test data sets (and, hence, hypergraphs). A more detailed description of each data set follows.

`yelp-dataset-1`. We built the first data set by collecting one million of randomly chosen reviews, from which we selected the businesses and the users to generate the hypergraph. Such selection also defined the total number of businesses and users involved, in other

words, the size of the hypergraph itself. We run our analysis on the largest connected component of the so-built hypergraph, removing isolated vertices and small components. More detailed information about the dimension of the network can be found in the following section.

`yelp-dataset-2`. Before generating this data set, we first analyzed the business categories distribution in the whole data set, where a category is a label describing the typology of the business, such as *Bars* or *Shopping*. Figure 7.2 (on the left) presents the business categories distribution plotted against the number of reviews associated with each category. As clearly visible from the plot, the most common business typology is *Restaurant*.

Figure 7.2 (on the right) focuses on the cuisine type distribution evaluated within the *Restaurant* macro-category. Each Yelp restaurant may offer several types of cuisine (e.g., Indian, Chinese, Asian fusion), but we labeled each business with a single *food category*, assigning the most frequent tag in the database. Namely, if a restaurant R had two genres A and B, but A was overall more frequent in the data set, we labeled R with A.

Based on this insight, we built the second data set considering only restaurants as a business category, attaching to each restaurant the label (selected from its categories set) representing the type of cuisine it offers according to the methodology just described.

Experiment 1: Rating Prediction

In this first experiment, the task performed was predicting the number of stars of a given business v, based on the information available in the local neighborhood of v. To solve this task, we adopted the naive approach of averaging the number of stars of all businesses related to v, according to the interactions defined by the relations *user-review-business*. Specifically, we applied this strategy on both (i) the hypergraph H defined in the previous paragraph and (ii) on the corresponding weighted two-section

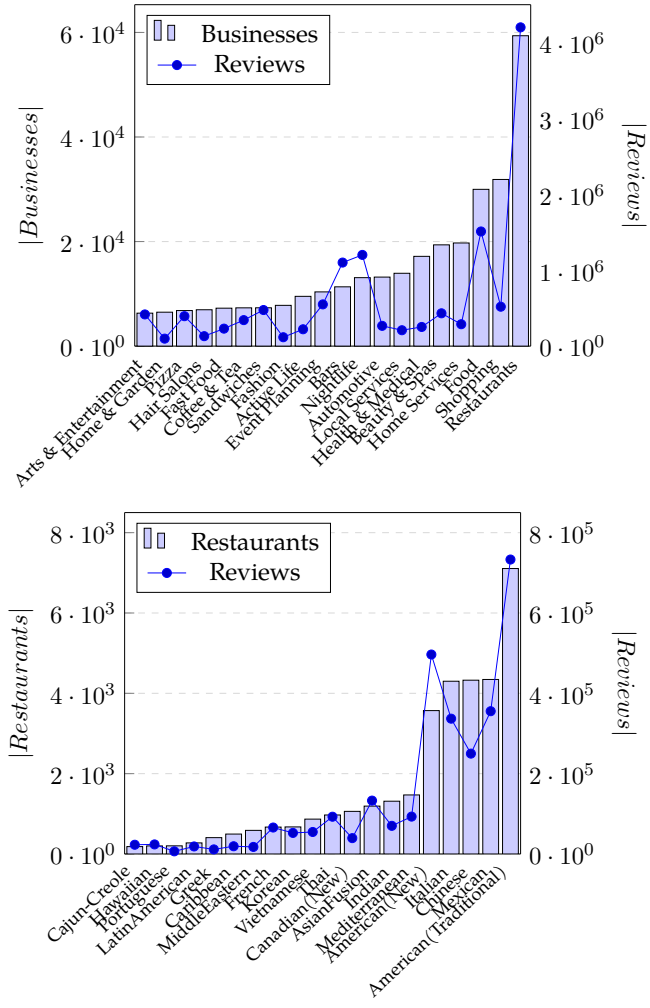


Figure 7.2: Businesses (left) and Restaurants (right) distribution together with the number of reviews associated with each category.

graph $[H]_2^w$, in which the weight of an edge (u, v) corresponds to the number of users that reviewed both u and v (that is, the number of hyperedges containing both u and v). More details follow.

Let us instantiate the described approach on the hypergraph H. For each business v, we first computed the average number of stars for all hyperedges containing v, excluding v itself. Simply put, this step corresponds to sum up the average rating given by the user associated with the hyperedge e. Then, the final prediction of the number of stars of v is obtained by dividing the values computed in the previous step by the degree of the vertex v. Formally,

$$s'_H(v) = \frac{1}{\deg(v)} \sum_{\substack{e \in H(v) \\ \kappa(e) > 1}} \left(\frac{1}{\kappa(e) - 1} \sum_{\substack{u \in e \\ u \neq v}} s(u) \right),$$

where $s(u)$ denotes the number of stars associated with u, $H(v)$ the family of hyperedges containing v, and $s'_H(v)$ the predicted value for v when modeling the underlying network with a hypergraph.

When modeling the Yelp.com review network using the weighted two-section graph $[H]_2^w$ of H, the predicted number of stars $s'_{[H]_2^w}(v)$ of a business v is the weighted average over the neighborhood of v. Formally,

$$s'_{[H]_2^w}(v) = \frac{\sum_{e=(v,u) \in E} s(u) w(e)}{\sum_{e=(v,u) \in E} w(e)},$$

where $w(e)$ denotes the weight of the edge e.

To compare the two network models, we computed the average error as follows:

$$err_x = \frac{\sum_{u \in V} |s(u) - s'_x(u)|}{|\mathcal{V}|},$$

where x indicates the typology of the network used (hypergraph or graph).

We performed this experiment on several instances of the `yelp-dataset-1`, varying the number of reviews used to build the corresponding hypergraphs. Specifically, we selected five subsets of increasing size equal

to 250k, 500k, 750k, and 1 million reviews. All hypergraphs resulted in having $209,393$ vertices, while the number of hyperedges ranged from $175,022$ to $432,381$. Figure 7.3 plots the error for both network models. Interestingly, we can note that the error using the weighted two-section graph is always more significant than the error obtained exploiting the hypergraph representation. We also performed the same experiment on the `yelp-dataset-2`, which generated a hypergraph with $35,466$ vertices and $1,133,890$ hyperedges. Similar to what happened with the first data set, the error for the two-section graph was always higher than the error for the hypergraph, reaching values 0.6 and 0.5, respectively.

Essentially, this experiment emphasizes the critical role played by high-order interactions that cannot be modeled using classic network science tools, like graphs. In this specific case, considering the voting pattern of each user singularly (and, thus, the data modeled by each hyperedge) conveys more valuable information resulting in a more accurate prediction of the actual rating of a business.

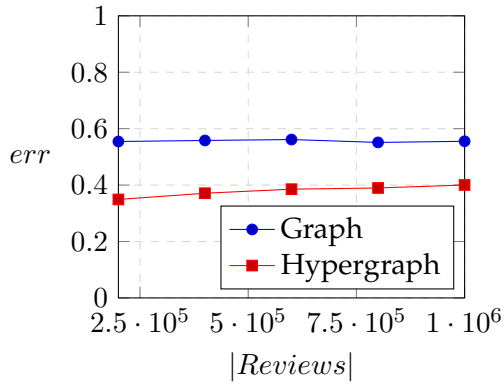


Figure 7.3: Average error of the rating prediction experiment performed on `yelp-dataset-1`, varying the number of reviews used to build the hypergraph and the corresponding two-section graph.

Experiment 2: Analyzing Review Patterns

In this second experiment, we focused on examining users' review patterns to understand which factors influence the chance that a given user reviews any two businesses and whether the hypergraphs induced by positive, neutral, and negative reviews are similar and to what extent. As in the first experiment, we compared the hypergraph model with the corresponding two-section graph, testing, in this case, their ability to detect the community structure of the underlying network. In other words, we were interested in finding a division of the vertex set into groups of businesses similar among themselves but dissimilar from the rest of the network and getting insights into the nature of their similarity.

Due to performance issues, we only ran the experiment on `yelp-dataset-2`, thus, restricting the set of businesses only to restaurants. We further split the reviews within the data set into five sets, according to the rating associated with them, ending up with $342{,}044$ **1**-star reviews, $281{,}307$ **2**-star reviews, $402{,}053$ **3**-star reviews, $791{,}068$ **4**-star reviews, and $1{,}188{,}558$ **5**-star reviews. We then built as many hypergraphs, one for each set. Henceforth, we will denote with H_i, for $i = 1, 2, \ldots, 5$, the hypergraph generated using the set of reviews having i stars and by G_i the corresponding two-section graph.

Table 7.2 reports the size of each hypergraph and the associated clique graph, along with information about the modularity value and the number of triangles evaluated over each graph. As expected, the number of edges quickly explodes according to the hyperedge size: the more restaurants a user has reviewed, the bigger the associated hyperedge is, and hence the bigger the number of edges in the clique graph is. In fact, for each hyperedge e, we have $(\kappa(e) \times (\kappa(e) - 1))/2$ edges. Interestingly, the number of edges and triangles in the clique graphs exhibits a "bell-shaped" trend as a function of the number of stars, with the mean value shifted towards 4-stars ratings.

To have a first insight on the nature of the so-built hyper-networks, we manually partitioned the five hypergraphs $H_{i \in [1,5]}$ according to different

Table 7.2: Size of each hypergraph $H_{i\in[1,5]}$ and the associated clique graph, along with the modularity value and the number of triangles evaluated for each graph.

Stars	H_i $(\lvert\mathcal{V}\rvert;\lvert E\rvert)$	G_i $(\lvert\mathcal{V}\rvert;\lvert\mathcal{E}'\rvert)$	G_i Modularity	G_i Triangles
1	(29,479; 244,671)	(29,479; 240,412)	0.6210	1,158,341
2	(28,055; 173,140)	(28,055; 484,527)	0.7173	6,491,497
3	(30,369; 177,792)	(30,369; 2,636,712)	0.6616	289,584,451
4	(32,987; 301,578)	(32,987; 4,384,044)	0.6857	404,709,664
5	(32,558; 590,320)	(32,558; 2,187,473)	0.6657	104,128,714

restaurant properties available in the Yelp.com data set. Specifically, we considered the following attributes: the location of the restaurant (*city*, and *state*), whether it sells *alcohol*, its *noise level*, whether it offers a *takeaway* service, and its food *category*. Following this process, the goodness of each partition, evaluated using the modularity measure, indicates whether a user tends to review restaurants having common characteristics.

Table 7.3 contains the modularity scores for the six different partitions evaluated. To compute the modularity, we used the approach presented by Kamiński et al. in [113] and implemented in SimpleHypergraphs.jl (see Section 7.3.2). The table shows that the modularity is generally stronger when using geographical information, such as cities or states, to partition the hypergraph. These higher values suggest that (*i*) Yelp users, who leave reviews, usually visit restaurants within the same city and that (*ii*) if the same person reviews restaurants in different cities, they are usually in the same state. Further, it is interesting to note that the 1-star reviews hypergraph H_1 broadly shows the strongest modularity values across all partitions. This outcome indicates the presence of a group of people who may have a stronger tendency to submit negative scores based on some ground truth property of a restaurant.

Starting from this first analysis, we then investigated the partition ob-

Table 7.3: Hypergraphs size for each $H_{i\in[1,5]}$ and modularity scores computed on the six different manually-evaluated ground truth partitions, conditioned on some properties of Yelp.com restaurants.

Stars	H_i ($\|\mathcal{V}\|$; $\|E\|$)	City	State	Alcohol	Noise Level	Take Out	Category
1	(29,479; 244,671)	0.8833	0.9562	0.8166	0.8104	0.8176	0.8163
2	(28,055; 173,140)	0.8582	0.9462	0.7744	0.7651	0.7731	0.7702
3	(30,369; 177,792)	0.8132	0.9226	0.7075	0.6940	0.6966	0.6965
4	(32,987; 301,578)	0.7812	0.9081	0.6573	0.6385	0.6419	0.6400
5	(32,558; 590,320)	0.8027	0.9145	0.6963	0.6797	0.6894	0.6841
ALL	(35,856; 950,488)	0.7500	0.8985	0.6162	0.5919	0.6013	0.5967

tained via a community detection algorithm. In this case, we used the "type of cuisine" (food *category*) of each restaurant as a ground truth. The ground truth partition was made up of 55 categories, of which the largest (American Traditional) comprised $7{,}107$ restaurants. As in the first experiment, we compared the communities obtained on both models (hypergraph and two-section graph) against the given ground truth partition to catch the best model in capturing that specific feature of the underlying network.

Several community detection algorithms have been proposed in the literature for graphs, and a comprehensive review can be found in [102]. For this experiment, we opted for the LP strategy proposed by Raghavan et al. [157], and we used the LP implementation provided by the Julia LightGraphs.jl library. As we needed an LP algorithm suitable for hypergraphs to have a fair comparison, we proposed a hypergraph-specific LP implementation, then integrated it into SimpleHypergraphs.jl (see Section 7.3.2).

To evaluate the correlation between two partitions, we used the *sum* variant of the *Normalized Mutual Information* (NMI) coefficient [183], a notion borrowed from information theory, which considers each partition as a probability distribution. In more detail, the sum variant of the NMI

coefficient is defined as

$$NMI(X,Y) = \frac{I(X,Y)}{H(X)+H(Y)},$$

where $I(X,Y)$ denotes the mutual information (i.e., the shared information between the two distributions X and Y, and $H(X)$ denotes the Shannon entropy (i.e., the information contained in the distribution) of X. The NMI coefficient holds several interesting properties: it is a *metric*, and it lies within a fixed range $[0,1]$. Specifically, it equals 1 if the partitions are identical, whereas it has an expected value of 0 if the two partitions are independent.

Results appear in Figure 7.4. Although, in general, the correlation values are not very high (the best result is 0.23 for H_5), the figure points out two interesting lines of discussion. First, in all five cases, the quality of partitioning provided by hypergraphs is always better than that provided by the corresponding two-section graphs. Moreover, results appear in the form of an "inverted bell shape", where the best results are given by the two external values. In a sense, this outcome suggests that excellent and low-star reviews can identify restaurants' genres much better than neutral comments. From a user perspective, this consideration may indicate the presence of a group of users usually visiting the same genre of restaurants and giving consistent ratings.

Some Performance Insights

We performed both experiments on a Linux Ubuntu 18.04 machine, equipped with an Intel i7 processor endowed with 8 cores, 16 GB of memory, and 256 GB SDD disk. We implemented the experiments using Julia $1.3.1$. With this configuration, Julia required about 117.68 seconds to load the Yelp data set in memory (about 9 GB).

In the following, we present the average performance in seconds, obtained running 10 times each experiment. In the first case study, we performed the forecasting algorithm on each hypergraph and the corresponding two-section graph. The completion time for the biggest network was

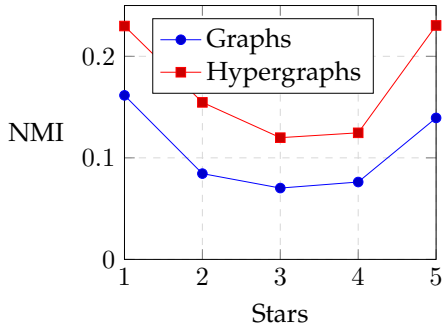


Figure 7.4: NMI values between the "type of cuisine" ground truth partition and the 10 partitions obtained running the LP algorithm on the five hypergraphs and the corresponding 2-section graphs. The experiment was run on `yelp-dataset-2`.

about 42 seconds, while the smallest one required 0.92 seconds. In the second use case, the average completion time for all scenarios was about 20 seconds. Specifically, we run both LP algorithms setting the maximum number of iterations to 100.

7.4.2 Mining and Modeling Social Relationships: the GoT Case

Another interesting line of inquiry is grasping the intricacies of a literary work or a movie to get insights into their narrative structure. Various works focused on finding common patterns across several plot lines [43], making sense of intricate character relationships [38], or just having fun trying to predict how the plot itself will evolve [170]. Usually, the character interaction network is modeled with a graph, where a vertex represents a storyline character, and an edge points out an interaction between two characters. Edges may also have different nature; for instance, they can express that the names (or aliases) of two characters appear within a certain number of words apart [2, 43], that a character A has spoken right after a character B, or that a character A and character B appear in a scene together [38]. The output graph is, commonly, an undirected, weighted

network.

A network built this way is an example of artificial collaboration networks, as most pairs of characters have cooperated or have been antagonists of one another [5]. A typical analysis carried on this kind of network is determining the most important characters, according to several centrality measures. Some vertices play a massive role in the network, either by having many connections or being strategically positioned to help connect distant parts of the network. Indeed, a character may be relevant or influential with different facets, and it is fundamental to interpret these quantities with respect to the underlying domain. Another question an exploratory analysis of networks of characters aims to answer is to capture which characters naturally belong together, forming coherent communities within the network. Investigating these behavioral patterns means looking for coherent sub-plotlines hidden in the network [37].

It is straightforward to see that a character interaction network built upon characters' co-occurrence in movie scenes can be naturally modeled with a hypergraph, where vertices are still associated with characters and hyperedges are related to scenes. In this case, the topology itself of the hypergraph allows us to easily find clusters of characters that commonly appear together within a movie or a TV series episode or season. To verify if hypergraphs can convey more information than a standard graph analysis approach, also in the case of collaboration networks, we have modeled and analyzed the Game of Thrones TV series characters co-occurrence. As for the Yelp use case (see Section 7.4.1), we compare the findings obtained by exploiting hypergraphs with the two-section graphs' results.

Tutorial Availability

A Jupyter Notebook replicating the analysis discussed in this chapter is available at the following GitHub public repository: `https://github.com/pszufe/SimpleHypergraphs.jl`.

The Game of Thrones TV Series Data Set

Game of Thrones [91] (GoT) is an American fantasy drama TV series created by D. Benioff and D.B. Weiss for the American television network HBO. It is the screen adaption of the series of fantasy novels *A Song of Ice and Fire*, written by George R.R. Martin. The series premiered on HBO in the United States on April 17, 2011, and concluded on May 19, 2019, with 73 episodes broadcast over eight seasons. With its 12 million viewers during season 8 and a plethora of awards — according to Wikipedia [1] — Game of Thrones has attracted record viewership on HBO and has a broad, active, and international fan base. The intricate world narrated by George R.R. Martin and scripted by Benioff and Weiss extend well beyond the boundaries of the traditional TV medium to create a deeply immersive entertainment experience [23]. This allows both the academic community and industries to study not only complex dynamics within the GoT storyline [38] but also how viewers engage with the GoT world on social media [11, 156, 158] or how the novel itself is a portrait of real-world dynamics [131, 147, 136, 138, 196].

This case study is based on the data set built by Jeffrey Lancaster on his GitHub repository *Game of Thrones Datasets and Visualizations* [103] (see Chapter 6). Specifically, we used the data describing season episodes, containing meta-information such as title, identification number, season, and description. Information about each scene within an episode is also

[1] `https://en.wikipedia.org/wiki/Game_of_Thrones`

Table 7.4: Number of episodes, scenes, and characters per GoT season.

Season	Episodes	Scenes	Characters
I	10	286	125
II	10	468	137
III	10	470	137
IV	10	517	152
V	10	508	175
VI	10	577	208
VII	7	468	75
VIII	6	871	66

reported. For each scene, start, end, location, and a list of characters are listed. Table 7.4 reports the same data shown in Chapter 6 with some basic information about the number of episodes, scenes, and characters per GoT season, as those values will represent the size of the hypergraphs analyzed.

Modeling GoT Using Hypergraphs

GoT characters' co-occurrences are well suitable to be studied via different levels of granularity. For instance, the hyper-network induced by co-occurrences can be modeled using three different hypergraphs, each reporting whether the GoT characters have appeared in the same season, the same episode, or the same scene together. A more detailed description follows.

- *Seasons.* A coarse–grained model is represented by the season hypergraph $H_s = (\mathcal{V}, E_s)$, where $\mathcal{V}$ represents GoT characters, and E_s represents the family of hyperedges in which each hyperedge is the set of characters appearing in a GoT season together. Figure 7.5 shows the hypergraph H_s using an Euler-based visualization (see §Hypergraph Visualization in Section 7.3.2).

- *Episodes×Season.* An intermediate-grained model is obtained by considering co-occurrence within episodes. In this case, we can build 8 different hypergraphs, $H_e^s = (\mathcal{V}^s, E_e^s)$, $s \in [1, 8]$, where s indicates a GoT season, $\mathcal{V}^s$ represents the GoT characters appearing in season s, and E_e^s is the family of hyperedges in which each hyperedge is the set of characters appearing in the episode e of season s. The 8 hypergraphs H_e^s are shown in Figure 7.6.

- *Scenes×Season.* A fine-grained model is obtained by considering co-occurrence within scenes. In this case, we can consider 8 different hypergraphs, $H_{sc}^s = (\mathcal{V}^s, E_{sc}^s)$, $s \in [1, 8]$, where s indicates a GoT season, $\mathcal{V}^s$ represents the GoT characters appearing in season s, and E_{sc}^s is the family of hyperedges in which each hyperedge is the set of characters appearing in the scene sc of season s.

In the case study presented in this chapter, we will focus on the Scenes×Season hypergraphs.

The Collaboration Structure of GoT

We performed a community detection task on the Scenes×Season hypergraphs and their corresponding two-section graphs. Running the community detection algorithm on such networks allows us to find coherent plotlines and, therefore, groups of characters frequently appearing in a scene together. In this experiment, our goal is to figure out whether and to what extent hypergraphs are able to capture characters' collaboration (and, generally speaking, any user-defined collaboration) with respect to graphs. Specifically, we ran the LP algorithms defined in §Community Detection in Section 7.3.2 and measured the quality of the solution obtained by computing the modularity score.

Results, described in Table 7.5, show that the solutions obtained for hypergraphs provide more communities than graphs. This pattern particularly emerges in the graphs describing the last two seasons, characterized by fewer characters.

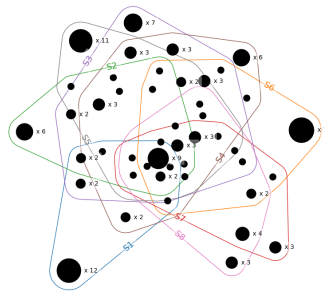
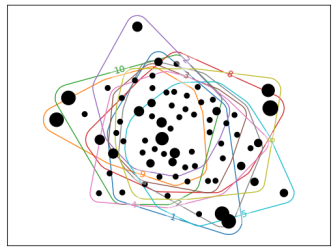
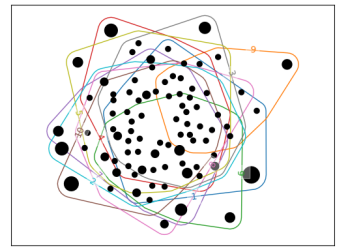


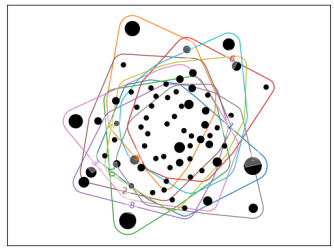
Figure 7.5: The GoT season hypergraph H_s defined by characters overlapping through seasons. Vertex labels indicate the number of vertices represented by a single aggregated vertex.



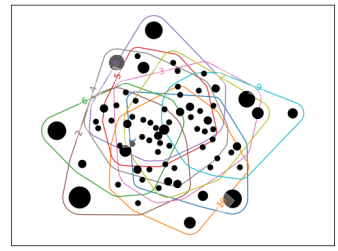
(a) H_e^1.



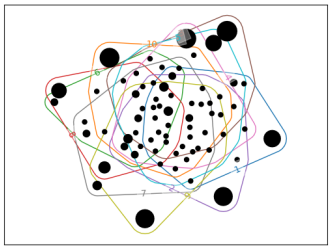
(b) H_e^2.



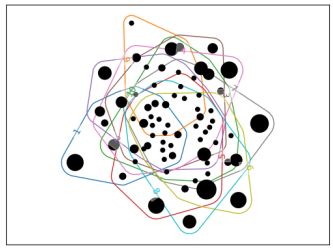
(c) H_e^3.



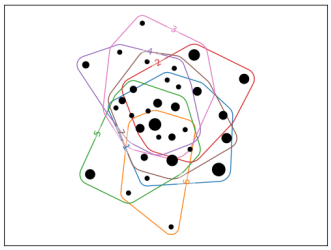
(d) H_e^4.



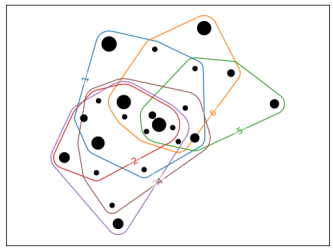
(e) H_e^5.



(f) H_e^6.



(g) H_e^7.



(h) H_e^8.

Figure 7.6: GoT characters overlap in the Episodes×Season hypergraphs.

To measure the difference between the two approaches, we pairwise computed the NMI between the obtained partitions. Table 7.6 shows NMI values, which reveal how the partitions are generally strongly related. An exception is made for the last two seasons, for which we can notice the worst results. We believe this outcome to be well justified to the nature of the last seasons' screenplay: fewer characters and high related plot-lines (see Figure 7.6h). Consequently, the hypergraphs induced by the co-occurrence network are characterized by a high degree for both vertices and hyperedges, and, by extension, the corresponding two-section graphs result in *quasi*-complete graphs. For this reason, the LP algorithm ran on so-built graphs cannot find out distinct communities.

Discussion on season 8. It is worth discussing the interesting facts revealed by the results of the 8th season. In the case of the two-section

Table 7.5: A comparison between hypergraph and graph capability on discovering characters' communities. For each hypergraph H^s_{sc} and its two-section graph $[H^s_{sc}]_2$, the table provides the number of communities $|C|$ and the corresponding modularity values m.

	H^1_{sc}	H^2_{sc}	H^3_{sc}	H^4_{sc}	H^5_{sc}	H^6_{sc}	H^7_{sc}	H^8_{sc}
H^s_{sc} $(\|C\|, m)$	$(12, .5359)$	$(15, .4511)$	$(15, .7028)$	$(18, .5527)$	$(22, .5794)$	$(19, .5781)$	$(11, .2159)$	$(8, .1536)$
$[H^s_{sc}]_2$ $(\|C\|, m)$	$(8, .3399)$	$(9, .6970)$	$(11, .7652)$	$(11, .6218)$	$(15, .7300)$	$(16, .7342)$	$(4, 2163)$	$(1, .0)$

Table 7.6: NMI values evaluated between the partitions obtained computing the communities on the Scenes×Season hypergraphs and their corresponding two-section graphs.

	S_1	S_2	$S3$	S_4	S_5	S_6	S_7	S_8
NMI	.82085	.83354	.92340	.87596	.88143	.88143	.69506	.0

graph, the label propagation algorithm reveals only one big community– the whole graph itself. However, in the case of hypergraphs, it can determine eight different communities. The following discusses the conflicting results obtained by providing a possible interpretation for the eight discovered communities by the label propagation algorithm run on the hypergraph model.

In more detail, three minor characters' communities, appearing only in a few scenes in the whole season, emerged from the (hyper)network structure. These communities are made up of: (i) the *Winterfell* boy — appearing only in the first episode; (ii) *Dirah*, *Craya*, *Marei* — seen only in the first episode; and (iii) *Eleanor* and her daughter — occurring only in the fifth episode. The algorithm performed on the scene co-occurrence hypergraphs correctly identifies background characters that do not contribute to the main storyline and are not strictly related to any main character.

Another two communities pinpoint key characters belonging to the two major alliances: the *north* versus the *south*. The south-alliance comprises *Cercei Lannister*, her counselor *Qyburn*, her guard *Gregor Clegane*, and her husband *Euron Greyjoy*, while the north-alliance is forged by Jon

Snow and *Daenerys Targaryen*, with her dragons. In particular, in the north-alliance community also appear two enemies that have been faced by *Jon* and *Daenerys*: the *Night King* (and his white walker soldiers) and Harry Strickland, captain of the sellsword *Golden Company*.

Two more communities can be labeled as north-allied: they contain a group of characters that consistently have interacted or fought together. Indeed, one group contains *Bran Stark* and *Theon Greyjoy* (who stands guard for him in the battle for Winterfell), Lyanna Mormont, and Wun Wun (they fought against in the battle for *Winterfell*), *Lord Varys*, *Davos Seaworth*, *Grey Worm*, and *Jorah Mormont*. The other community includes *Arya* and *Sansa Stark*, *Samwell Tarly* and his partner *Gilly*, *Brienne of Tarth* and *Jamie Lannister*, *Tyrion*, *Tormund*, *Missandei*, *Melisandre*, and *Sandor Clagane*, among few others.

The algorithm also discovers a community related to the sub-plotline of *Yara Greyjoy* and some lords loyal to her: after having been saved by her brother *Theon*, she leaves the stage to claim her land. She reappears only in the last episode of the season, together with the other main characters. In this group, we can also find two royal background characters - *Edmure Tully* and *Robin Arryn* - that do not contribute to the development of the main plotline and only appear in the last episode.

Which Are the Most Important Characters?

Identifying truly important and influential characters in a vast narrative like GoT may not be a trivial task, as it depends on the considered level of granularity. In these cases, the main character(s) in each plotline is referred to with the term *fractal protagonist*(*s*) to indicate that the answer to "who is the protagonist" depends on the specific plotline [37]. In this section, following the same methodology of previous experiments, we evaluate GoT characters according to the degree and betweenness centrality metrics, exploiting the Scenes×Season hypergraphs and the corresponding two-section graphs.

Degree centrality. Loosely speaking, this metric gives information about the number of interactions of a vertex. If we consider a hypergraph H^s_{sc}, the degree centrality of a vertex is the number of scenes in a given season s a character v appears in. In other words, we are enumerating the number of hyperedges in which the vertex is contained. Formally,

$$C_D(v) = |H(v)| = deg(v).$$

Analogously, the degree centrality of a character v, on the associated two-section graph $G = [H^s_{sc}]_2$, represents the number of characters they played with during season s.

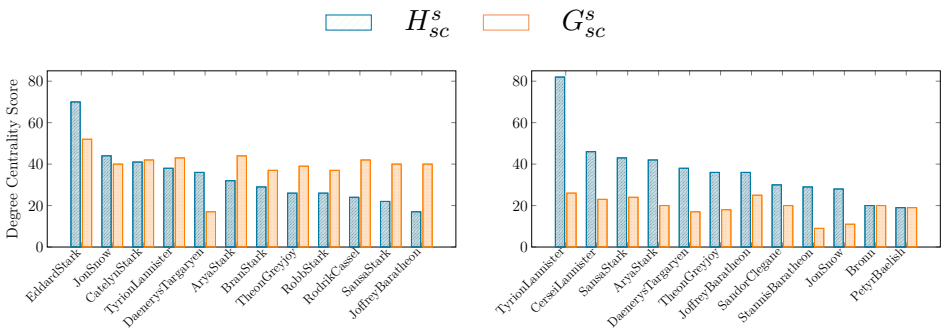
Figure 7.7 shows the degree centrality scores of 12 GoT characters per season, evaluated on the Scenes×Seasons hypergraphs H^s_{sc} and the corresponding two-section graphs G^s_{sc}. Characters are sorted according to their degree centrality in the hypergraph. It is worth noting that we did not normalize the degree centrality values as for the hypergraph this would mean normalizing $C_D(v)$ by $2^n - 1$, considering all possible hyperedges in which a vertex can be contained; thus, leading to infinitesimal scores. As the unnormalized centrality scores are not directly comparable, we will only consider the resulting character ranking.

The first outcome to comment on is the almost null consensus between the same metric evaluated on the two structures. Although interesting, this result resembles the different meanings that the concept of *centrality* assumes. Evaluating the degree centrality on the Scenes×Seasons hypergraphs means ranking GoT characters based on their screen appearance, which usually is a good metric to estimate the importance of a character within a TV series storyline. Hence, each plot gives us insights about which characters sub-plotline each season focuses. In contrast, evaluating the Scenes×Seasons two-section graph means ranking GoT characters according to how many other characters they have interacted with.

The second point to notice is that the hypergraph degree centrality tends to be more coherent with the real importance of each character in the actual storyline. In particular, we can mention a few cases in which

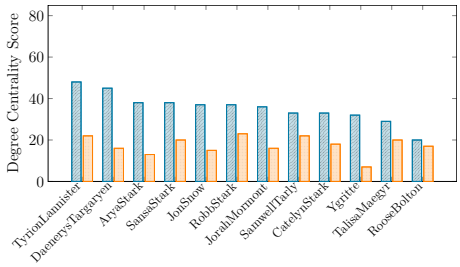
considering the graph degree centrality may also be misleading in capturing the relevance of GoT characters. For instance, Daenerys Targaryen interacts with only a few characters in the first season and, consequently, has a low degree centrality in the two-section graph. However, the emphasis of her role becomes more evident when considering the hypergraph. A similar situation happens for Jon Snow and his close friend Eddison Tollett in Season 5. Looking at the graph degree centrality, it seems that the latter has a similar role in the GoT vicissitudes as the protagonist, while the hypergraph degree centrality reveals a completely reversed perspective. Clearly, this trend directly results from the meaning of centrality in the Scenes×Seasons hypergraphs.

Betweenness centrality. We also investigated the importance of the characters evaluating the betweenness centrality metric of hypergraph vertices. The betweenness centrality measures the centrality of a node by computing the number of times that a node acts as a bridge between the other two vertices, considering the shortest paths between them.

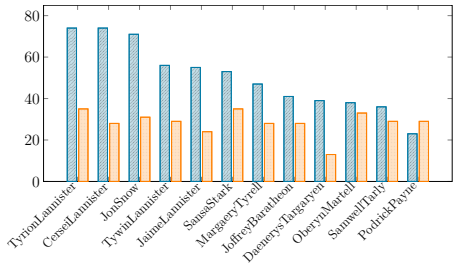


(a) Season 1.

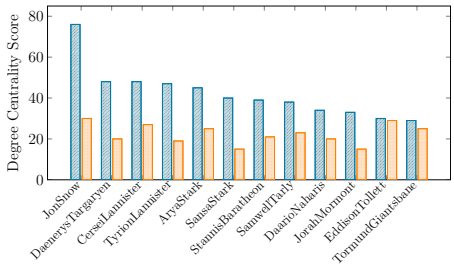
(b) Season 2.



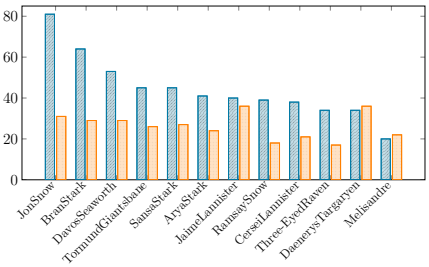
(c) Season 3.



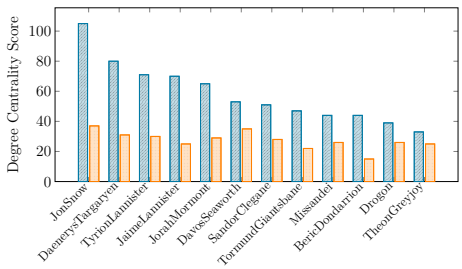
(d) Season 4.



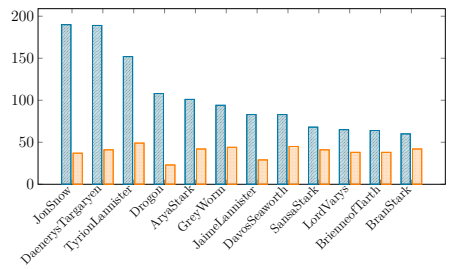
(e) Season 5.



(f) Season 6.



(g) Season 7.



(h) Season 8.

Figure 7.7: Degree centrality scores of GoT characters per each season, evaluated on the Scenes×Seasons hypergraphs H^s_{sc} and the corresponding two-section graph G^s_{sc}.

Based on the concept of s-node-shortest-path between two different vertices u and v (see §Paths and Connected Components in Section 2.1) implemented in the Python library HyperNetX [107] and later formalized by Aksoy et al. [4], we developed the notion of s-betweenness centrality.

The underlying idea of this centrality metric was to compute the betweenness centrality considering a path made by more robust connections (or interactions), which should be more precise indicators of the importance of the characters in each season. Formally, the s-betweenness centrality is defined as

$$C_B^s(v) = \sum_{\substack{x,y \in \mathcal{V} \setminus \{v\} \\ x \neq y}} \frac{\sigma_{xy}^s(v)}{\sigma_{xy}^s},$$

where $\sigma_{xy}^s(v)$ is the number of the s-node-shortest-paths between two vertices x and y that pass through v, while σ_{xy}^s is the total number of s-node-shortest-paths between x and y. It is worth noting that C_B^s generalizes the definition of betweenness centrality. Using $s = 1$, C_B^1 reduces to the classical betweenness centrality defined on graphs.

We evaluated the betweenness centrality scores on the Scenes×Seasons hypergraphs varying s in $[1, 3]$. Figure 7.8 shows the results from Season 1 to Season 8. In this case, each plot shows a different number of characters as we considered the ten highest-ranked characters for each metric and sorted them according to C_B^1 to highlight the differences in the three metrics adequately. For each season, we measured the Pearson correlation to evaluate the ranking agreement.

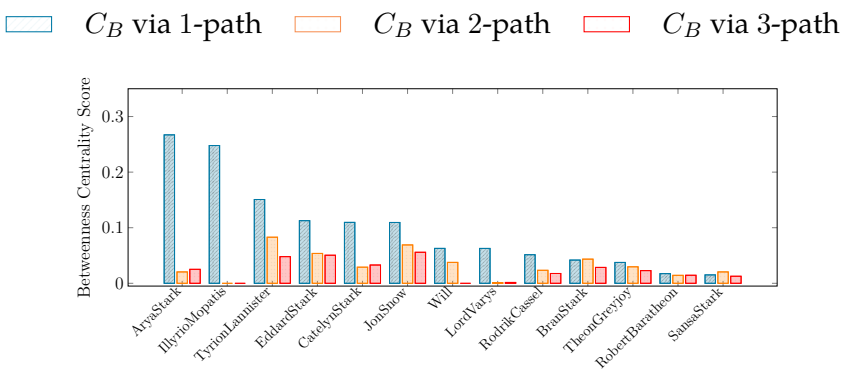
Table 7.7 reports the correlation scores. Generally, we can observe a higher correlation between C_B^2 and C_B^3, while a lower correlation between C_B^1 and C_B^3, although with some exceptions. Season 4 shows the highest concordance among the three metrics (see Figure 7.8d), Season 2 the lowest (see Figure 7.8b).

The interesting outcome to note in this analysis is that the C_B^2 is able to give more valuable insights about the position each character plays in the GoT storyline. For instance, a GoT fan knows that Eddard Stark and Tyrion Lannister rather than Arya Stark or Illyrio Mopatis tie together with social interactions House Stark and House Lannister in Season 1. Further, Illyrio Mopatis disappear when considering (2 and 3)-node-shortest-paths. Another interesting example is visible in Season 6. Here C_B^2 and C_B^3 highlight Jamie Lannister and Brienne of Tarth as key vertices in the hypergraph

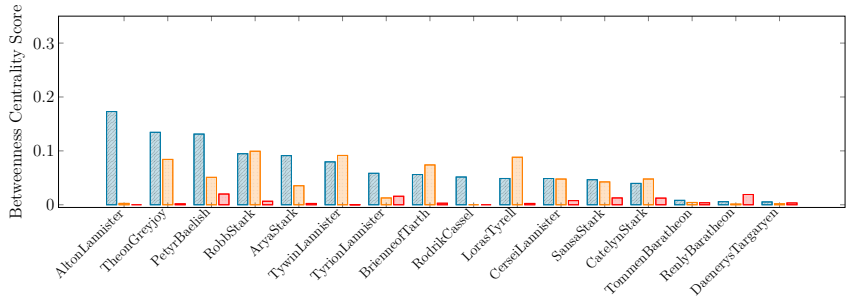
Table 7.7: Pearson correlation values for s-betweenness centrality using $\{1, 2, 3\}$-path.

	$\mathbf{S}_1$	$\mathbf{S}_2$	$\mathbf{S}_3$	$\mathbf{S}_4$	$\mathbf{S}_5$	$\mathbf{S}_6$	$\mathbf{S}_7$	$\mathbf{S}_8$
$\rho(C_B^1, C_B^2)$	-0.008	0.303	0.270	0.800	0.239	0.539	0.297	0.439
$\rho(C_B^1, C_B^3)$	0.117	-0.185	0.254	0.796	0.092	0.440	0.214	0.185
$\rho(C_B^2, C_B^3)$	0.837	-0.177	0.049	0.800	0.654	0.839	0.820	0.415

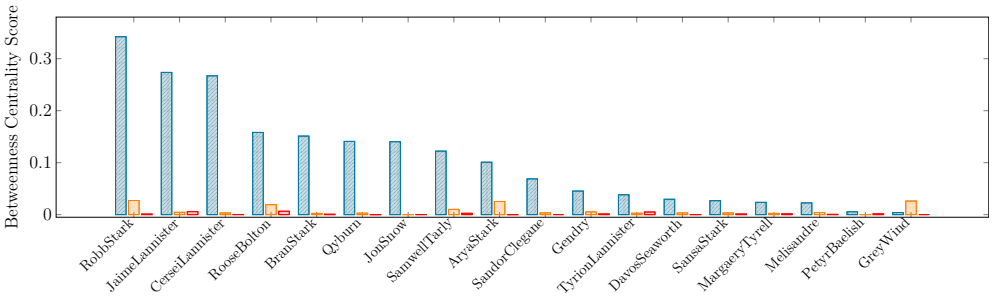
and, in fact, they are among the characters who travel and negotiate the most in that season. This information is hidden if looking only at the ranking based on C_B^1.



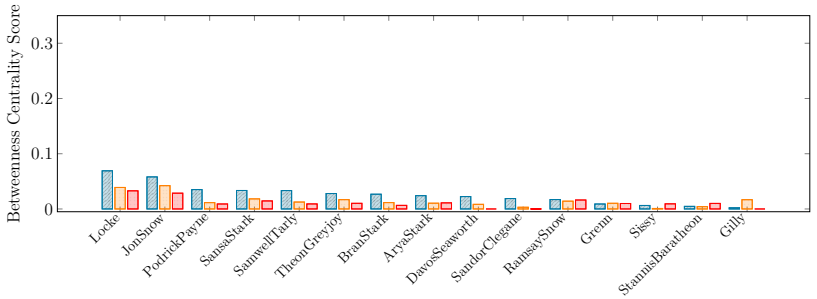
(a) Season 1.



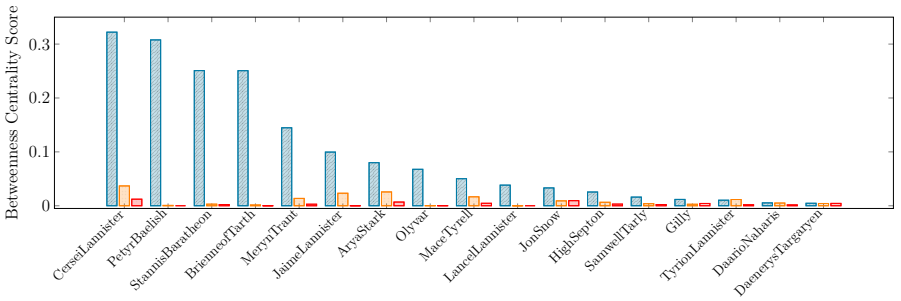
(b) Season 2.



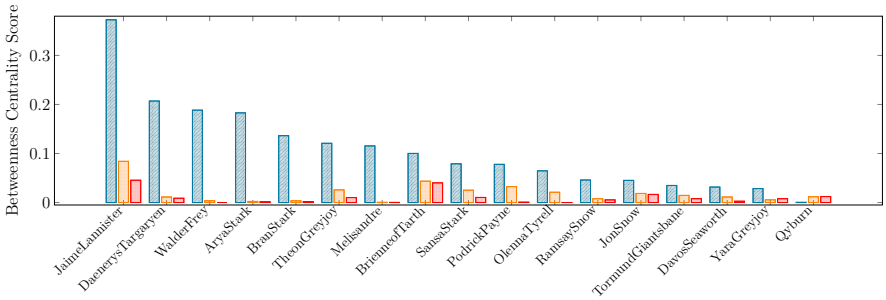
(c) Season 3.



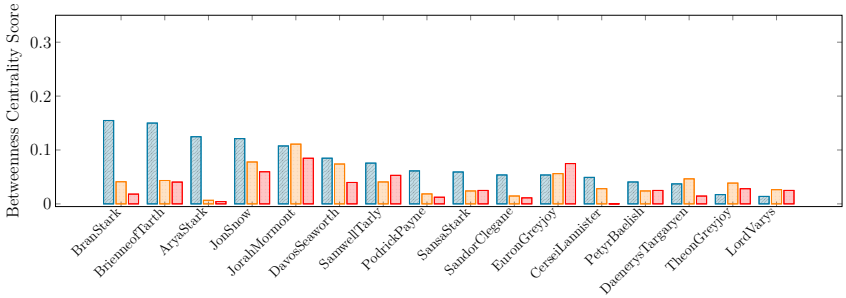
(d) Season 4.



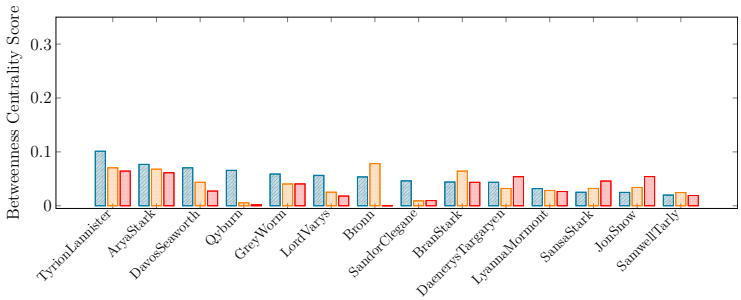
(e) Season 5.



(f) Season 6.



(g) Season 7.



(h) Season 8.

Figure 7.8: Betweenness centrality scores of GoT characters per each season, evaluated on the Scenes×Seasons hypergraphs H^s_{sc} and the corresponding two-section graph G^s_{sc}.

7.5 Remarks

Hypergraphs are a natural generalization of graphs and provide a much richer structure than their well-known graph counterparts, suitable for modeling many natural phenomena involving group-based interactions, such as collaborative activities. Working with hypergraphs requires software libraries specifically designed to perform operations directly on these high-order structures, from basic algorithms to more complex tasks. To address this need, this chapter presented SimpleHypergraphs.jl, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing. Currently, SimpleHypergraphs.jl provides a wide range of functionalities to build and explore hypergraphs. The library SimpleHypergraphs.jl:

- Ensures flexible interoperability with both the existing Julia ecosystem and the classical graph manipulation framework, providing twofold integration with both the Julia standard matrix type and the LightGraphs.jl package;
- Provides APIs representing a hypergraph $H = (\mathcal{V}, E)$ as an $n \times m$ matrix, and each entry (i, j) denotes the weight of the vertex i within the hyperedge j;
- Internally stores a hypergraph as a sparse array, and its data in a redundant format, using two separate hashmap structures for rows and columns. Further, the library provides several constructors and enables attaching metadata values of arbitrary type to both vertices and hyperedges;
- Supports:
  - the automatic generation of random hypergraphs either with or without any specific structural constraints;
  - several accessing and manipulating functions;
  - hypergraph to graph transformations;

  - serialization mechanisms;
  - centrality and community detection functionalities;
  - hypergraph visualization.

The development of this library is a joint project with the Warsaw School of Economics (Warsaw, Poland) and the Ryerson University (Toronto, Canada). Today, SimpleHypergraphs.jl is the referring Julia library to model, manipulate, and visualize hypergraphs, and it stands as a vehicle to enable current and future hypergraph research.

The second part of the chapter discussed two case studies with a two-fold objective.

- First, they demonstrate how it is possible to exploit SimpleHypergraphs.jl to perform via hypergraphs standard graph analysis in the network science landscape. In the first case study, we analyzed user review activities from the social network Yelp.com in the tasks of rating prediction and community detection. The second case study explored the communities induced by the scene co-occurrence hypernetwork of the GoT TV series and GoT characters' centrality features.
- Second, they compare hypergraphs with their corresponding graph representation to explore whether high-order structures convey more information in addressing specific tasks. The outcomes obtained in both scenarios suggest that the hypergraph structure seems to improve the analysis of high-order networks abstracting the many-to-many interactions defined by the underlying structure of the data. In particular, hypergraphs recognized specific Yelp users' review patterns better than the corresponding two-section graphs, and, similarly, those structures better identified GoT sub-plotlines and crucial characters.

Hypergraphs have become the subject of a growing literature only in the last years. However, more effort has to be put into understanding if the advantage of preserving a more detailed relationship structure justifies

the need for a more complex data structure and, as a consequence, more complex underlying algorithms. These two simple scenarios represent a small step in this direction.

PART

III

Social Influence Diffusion in High-order Networks

Summary

8

Influence Maximization in Hypergraphs

> Nothing happens until something moves.
>
> Albert Einstein

In short

This chapter presents the second central contribution of this dissertation: the formal definition of a high-order diffusion process, the generalization of a well-known graph problem to hypergraphs, and a set of heuristics to tackle it. Specifically, we first introduce the motivation behind this line of study (see Section 8.1) and discuss a new linear threshold (LT) high-order diffusion model that mimics real-world social dynamics, where individuals influence the group they belong to, but - in turn - the group itself influences their choices (see Section 8.2). We then introduce the formal definition of the Target Set Selection problem on hypergraphs (TSSH), a key algorithmic question in information diffusion

research, whose goal is to find the smaller set of vertices that can influence the whole network according to the diffusion model defined (see Section 8.2). Since the TSSH problem NP-hard, we describe four heuristics to address it (see Section 8.3), and we further review how our work differentiates from the state-of-the-art (see Section 8.4). We finally describe the experiment setting (see Section 8.5) and provide an extensive evaluation on random and real-world networks (see Section 8.6).

The work described in this chapter has been presented in the following articles:

- A. Antelmi, C. Cordasco, C. Spagnuolo, P. Szufel. Information Diffusion in Complex Networks: A Model Based on Hypergraphs and Its Analysis. In: *Algorithms and Models for the Web Graph*, pages 36-51, Cham, 2020. Springer Int. Publishing.

- A. Antelmi, C. Cordasco, C. Spagnuolo, P. Szufel. Social Influence Maximization in Hypergraphs. *Entropy*, 2021.

8.1 Motivation

Throughout the previous chapters of this dissertation, we have seen that hypergraphs may represent all kinds of high-order relations that can occur to our minds. In the specific context of social influence diffusion, hypergraphs can help us in modeling groups of people representing, for instance, real-world friend circles or online communities. Groups play a critical role in society, and understanding people's groups is fundamental to revealing personal behaviors [75] as these are heavily influenced by the crowd [69]. Groups can traditionally form offline but, with the advent of online social networks, many offline group decision processes switched to online (e.g., marketing strategies) [198]. As a consequence, the phenomenon of social influence assumed even more importance given the exponential number of people a user can reach through them. Social influence involves intentional and unintentional efforts to change another person's beliefs, attitudes, or behavior [80]. In both online and offline con-

texts, a person may influence other individuals for different reasons. As a domino cascade, such influence impacts not only those people but also propagates through the social network, with varying degrees of effectiveness. This *word-of-mouth* propagation has been shown to be a powerful tool in different applications (see Chapter 4). Hence, modeling diffusion phenomena conveyed by high-order interactions is crucial to correctly resemble these group dynamics happening in the real world.

8.2 Social Influence Diffusion in Hypergraphs

In this section, we discuss a new influence diffusion process on hypergraphs, generalizing the LT model proposed by Kempe et al. in their seminal work on influence maximization [117] (see Chapter 4). We further present the formal definition of the TSS problem on hypergraphs.

8.2.1 A Dynamic Social Influence Diffusion Process

When it comes to making a choice, groups can exert influence on their components, but also vice versa happens [154]. We embedded this idea into a dynamic social influence diffusion process on hypergraphs, where both vertices (people) and hyperedges (groups) can be influenced. This process models the influence propagation iteratively from vertices to hyperedges and from hyperedges to vertices so that it is possible to manage non-binary relations (as hyperedges can have any size), mimicking the real-world process. As friends can differently influence individuals, each neighbor of a vertex v in the hypergraph can have a different impact on influencing v itself, depending on the number and the size of the hyperedge shared with v. Further, by modifying the thresholds associated with hyperedges, it is possible to model different scenarios.

Thus, the diffusion process evolves in discrete steps. Loosely speaking, in the beginning, all vertices in a given set $\mathcal{S} \subseteq \mathcal{V}$ are considered active (or influenced), while all hyperedges are initially non-influenced. Then, at each iteration:

1. All hyperedges containing enough active vertices are added to the

set of influenced hyperedges;

2. All vertices contained in enough active hyperedges are added to the set of influenced vertices.

The process ends when no new vertices become active in two successive iterations.

In the following, we provide a formal definition of the process, using the incidence graph $I(H)$ associated to a hypergraph H.

Let $H = (\mathcal{V}, E)$ be a hypergraph and $I(H) = (\mathcal{V} \cup E, \mathcal{F})$ the associated incidence graph. Let $t_{\mathcal{V}} : \mathcal{V} \to [0, m]$ and $t_E : E \to [0, n]$ be two functions assigning thresholds to vertices and hyperedges, respectively. For each vertex $v \in \mathcal{V}$ ($e \in E$), the value $t_V(v)$ ($t_E(e)$) quantifies how hard it is to influence the vertex v (hyperedge e), in the sense that easy-to-influence elements of the network have *low* threshold values, and hard-to-influence elements have *high* threshold values. The same comment also holds for hyperedges. Let $t(\cdot)$ be a threshold function which assigns to each vertex $v \in I(H)$ the threshold value of the corresponding element of H, according to the following rule

$$t(v) = \begin{cases} t_{\mathcal{V}}(v), \text{ if } v \in \mathcal{V} \\ t_E(v), \text{ if } v \in E \end{cases} .$$

Definition 8.1 : Social Influence Diffusion Process in Hypergraphs

Let $I(H) = (\mathcal{V} \cup E, \mathcal{F})$ be a incidence graph associated to a hypergraph $H = (\mathcal{V}, E)$ with threshold function $t : (\mathcal{V} \cup E) \longrightarrow \mathbb{N}$.

An *information diffusion process* in $I(H)$, starting with a seed $\mathcal{S}$, is a

sequence of vertices subsets

$$I[\mathcal{S}, 0] \subseteq I[\mathcal{S}, 1] \subseteq \ldots \subseteq I[\mathcal{S}, \ell] \subseteq \ldots \subseteq \mathcal{V} \cup E,$$

with $I[\mathcal{S}, 0] = \mathcal{S}$, and such that for all $\ell > 0$,

$$I[\mathcal{S}, \ell] = I[\mathcal{S}, \ell - 1] \bigcup \Big\{ v \in \mathcal{V} \cup E : |N(v) \cap I[\mathcal{S}, \ell - 1]| \geq t(e) \Big\},$$

where $N(v)$ denotes the neighborhood of v.

The process ends at the first iteration ℓ such that $I[S, \ell - 1] = I[S, \ell]$.

We denote the final influenced sets as $I_{\mathcal{V}}[\mathcal{S}] = I[\mathcal{S}, \ell - 1] \cap \mathcal{V}$ and $I_E[\mathcal{S}] = I_E[\mathcal{S}, \ell - 1] \cap E$. We indicate the above information diffusion process on $I(H)$ with

$$I_V[\mathcal{S}], I_E[\mathcal{S}] = \Phi(H, \mathcal{S}, t_{\mathcal{V}}, t_E),$$

where, $I_{\mathcal{V}}[\mathcal{S}] \subseteq V$ is the set of influenced vertices and $I_E[\mathcal{S}] \subseteq E$ is the set of influenced hyperedges.

The diffusion process is irreversible; once a vertex or hyperedge is influenced, it remains in the active state until the end of the process. Further, we can note that, since $I(H)$ is a bipartite graph, the process proceeds by adding vertices corresponding to the hyperedges of H during odd iterations and vertices corresponding to vertices of H during even iterations (see Figure 8.2).

Definition 8.2 : Target Set

A *target set* for $H = (\mathcal{V}, E)$ is a seed set $\mathcal{S} \subseteq \mathcal{V}$ that will eventually influence all vertices, i.e., $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$. It is worth noting that a target set does not necessarily influence all hyperedges.

Example. Consider the bipartite incidence graph $I(H)$ associated to the hypergraph $H = (\mathcal{V}, E)$ in Figure 8.1, with vertex and hyperedge thresholds given in Figure 8.2. Starting with the initial seed set $\mathcal{S} = \{v_1, v_4\}$, the information diffusion process evolves as follows (see Figure 8.2):

$$
\begin{aligned}
&\textit{Step } 0: \quad I[\mathcal{S}, 0] = S = \{v_1, v_4\} \\
&\textit{Step } 1: \quad I[\mathcal{S}, 1] = \{v_1, v_4, e_2\} \\
&\textit{Step } 2: \quad I[\mathcal{S}, 2] = \{v_1, v_4, e_2, v_3\} \\
&\textit{Step } 3: \quad I[\mathcal{S}, 3] = \{v_1, v_4, e_2, v_3, e_3\} \\
&\textit{Step } 4: \quad I[\mathcal{S}, 4] = \{v_1, v_4, e_2, v_3, e_3, v_5\} \\
&\textit{Step } 5: \quad I[\mathcal{S}, 5] = \{v_1, v_4, e_2, v_3, e_3, v_5, e_1\} \\
&\textit{Step } 6: \quad I[\mathcal{S}, 5] = \{v_1, v_4, e_2, v_3, e_3, v_5, e_1, v_2\}
\end{aligned}
$$

As $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$, $\mathcal{S}$ is a target set for H.

(a) Hypergraph H.

(b) Bipartite graph $I(H)$.

Figure 8.1: A hypergraph $H = (V, E)$ with 5 vertices and 3 hyperedges and the bipartite incidence graph $I(H)$ associated to H (the five vertices appear on the left and the three hyperedges appear on the right).

8.2.2 The Target Set Selection (TSS) Problem on Hypergraphs

In this dissertation, we address the TSS problem on hypergraphs, whose objective is finding the minimum target set that will eventually influence the whole network. Formally, given a hypergraph $H = (\mathcal{V}, E)$ with threshold functions $t_{\mathcal{V}} : \mathcal{V} \longrightarrow \mathbb{N}$ and $t_E : E \longrightarrow \mathbb{N}$, we want to find a seed set $\mathcal{S} \subseteq \mathcal{V}$ of minimum size such that $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$.

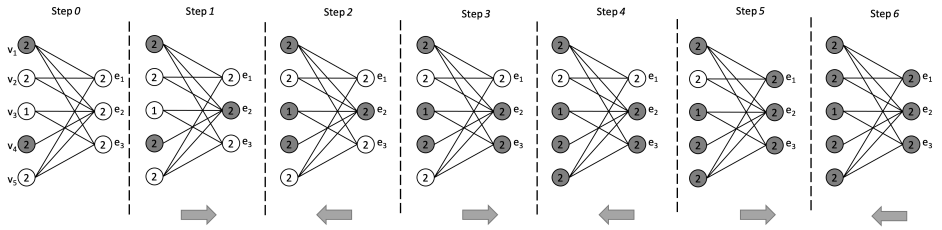


Figure 8.2: An example of influence diffusion process on $I(H)$ associated to $H = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{e_1, e_2, e_3\}$. Influenced vertices appear in gray. The arrow at the bottom of each step indicates the direction of the diffusion process: (left-to-right) vertices influence hyperedges; (right-to-left) hyperedges influence vertices.

Definition 8.3 : TSS problem on hypergraphs (TSSH)

Instance: $H = (\mathcal{V}, E)$, thresholds $t_{\mathcal{V}} : \mathcal{V} \to \mathbb{N}_0$ and $t_E : E \to \mathbb{N}_0$.
Problem: Find a seed set $\mathcal{S} \subseteq \mathcal{V}$ of minimum size such that $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$.

The TSSH problem generalizes the TSS problem on graphs, studied by Chen [54] and defined as follows:

> Given a graph $G = (\mathcal{V}, E)$ and fixed arbitrary thresholds $t(v)$, for each $v \in \mathcal{V}$, find a set of minimum size that eventually influences all vertices in G.

As any graph is a 2-uniform hypergraph, TSSH reduces to TSS by fixing $t_{\mathcal{V}}(v) = t(v)$ for each $v \in \mathcal{V}$ and $t_E(e) = 1$ for each $e \in E$. Chen proved a strong inapproximability results on TSS, showing that even a non trivial approximation of the problem is unlikely tractable. Since TSSH is a generalization of TSS, Chen's result also applies to TSSH.

It is also worth mentioning that having hyperedge thresholds $t_E(e) \geq 1$ on the TSSH problem does not imply that spreading influence is more

difficult in hypergraphs than in graphs since the diffusion process is ruled by both the thresholds and the size of the hyperedges. For instance, let us assume that all vertices have a threshold 2 and the existence of two hyperedges with a threshold 2, whose intersection contains 100 vertices. In this case, two influenced vertices in the intersection of the two given hyperedges can influence the other 98 vertices.

8.3 Finding the Minimum Target Set on Hypergraphs

In this section, we describe four greedy-based heuristics to address the *TSSH* problem defined in Section 8.2.2. Specifically, we first detail three additive solutions and then introduce a subtractive approach. We also define an optimization strategy, which can be used as a refinement of the seed set evaluated by each heuristic. We indicate the diffusion process on H as $\Phi(H, \mathcal{S})$, denoting with $I_{\mathcal{V}}[\mathcal{S}] \subseteq \mathcal{V}$ and $I_E[\mathcal{S}] \subseteq E$ the final sets of influenced vertices and influenced hyperedges, respectively.

8.3.1 Additive Heuristics

As their name suggests, additive heuristics adopt an additive approach to select the target set, adding the best vertices according to a given measure. The basic idea behind a dynamic implementation of these procedures is that first, the best candidate is added to the seed set, and then the hypergraph is pruned from all influenced vertices and hyperedges. In the following, we describe a static additive procedure (used as a baseline in our experiments) and two dynamic additive heuristics.

StaticGreedy

A trivial greedy strategy consists of selecting the vertices with the highest degree as a seed set. After sorting the vertices in descending order by their degree, the heuristic applies a binary search procedure to compute the target set $\mathcal{S}$. This heuristic is referred to as static since the vertex degree is never updated. Algorithm 1 details the pseudo-code of this procedure.

Algorithm 1 StaticGreedy$(H = (\mathcal{V}, E), t_{\mathcal{V}}, t_E)$

Let $\boldsymbol{\sigma}(\mathcal{V})$ be the list of vertices in non-ascending order of degree.
left $\leftarrow 1$, right $\leftarrow |\mathcal{V}|$
while left $<$ right **do** ▷ Binary Search
    mid $\leftarrow \left\lceil \frac{\text{left+right}}{2} \right\rceil$
    $I_{\mathcal{V}}[\mathcal{S}], I_E[\mathcal{S}] \leftarrow \Phi(H, \boldsymbol{\sigma}_{\leq \text{mid}}, t_{\mathcal{V}}, t_E)$ ▷ $\boldsymbol{\sigma}_{\leq i}$ is the sublist containing the first i vertices of $\boldsymbol{\sigma}(\mathcal{V})$
    if $I_{\mathcal{V}}[\mathcal{S}] \neq \mathcal{V}$ **then**
        left $\leftarrow$ mid
    else
        right $\leftarrow$ mid $- 1$
return $\mathcal{S} \leftarrow \boldsymbol{\sigma}_{\leq \text{left}}$

DynamicGreedy

In this heuristic, all vertices are initially added to the candidate set $\mathcal{U}$. At each iteration, the vertex having the highest degree is added to the target set $\mathcal{S}$ and removed from $\mathcal{U}$. At this point, the strategy simulates the diffusion process and prunes all influenced hyperedges from the hypergraph. Then, the residual degree $\delta(v)$ of each vertex v is updated accordingly, and the process is iterated until all vertices are influenced. Algorithm 2 details the pseudo-code of this procedure.

DynamicGreedy$_{[H]_2}$

This third heuristic is equivalent to $DynamicGreedy$, but it works on the clique-expansion $[H]_2$ of the input hypergraph H (see Section 2.2.2). Algorithm 3 details the pseudo-code of this procedure.

Algorithm 2 DynamicGreedy$(H = (\mathcal{V}, E), t_{\mathcal{V}}, t_E)$

$\mathcal{S} \leftarrow \emptyset, \mathcal{U} \leftarrow \mathcal{V}, E' \leftarrow E$
for $u \in \mathcal{U}$ **do**
    $\delta(u) \leftarrow deg(u)$
while $\mathcal{U} \neq \emptyset$ **do**
    $v \leftarrow \arg\max_{u \in \mathcal{U}} \ \delta(u)$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{v\}$
    $I_{\mathcal{V}}[\mathcal{S}], I_E[\mathcal{S}] \leftarrow \Phi(H, \mathcal{S}, t_{\mathcal{V}}, t_E)$
    if $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$ **then**
        return $\mathcal{S}$
    $E' \leftarrow E \setminus I_E[\mathcal{S}]$
    for $u \in \mathcal{U}$ **do**
        $\delta(u) \leftarrow |E(u) \cap E'|$ ▷ $\delta(u)$ denotes the degree of u in $H = (\mathcal{V}, E')$
return $\mathcal{S}$

Algorithm 3 DynamicGreedy$_{[H]_2}(H(\mathcal{V}, E), t_{\mathcal{V}}, t_E)$

$\mathcal{S} \leftarrow \emptyset, \mathcal{U} \leftarrow \mathcal{V}, E' \leftarrow E$
$[H]_2 \leftarrow$ TwoSection$(H(\mathcal{V}, E))$ ▷ The function TwoSection(-) transform H in its corresponding clique graph
while $\mathcal{U} \neq \emptyset$ **do**
    $v \leftarrow \arg\max_{u \in \mathcal{U}} \ deg_{[H]_2}(u)$ ▷ $deg_{[H]_2}(u)$ denotes the degree of u in $[H]_2$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{v\}$
    $I_{\mathcal{V}}[\mathcal{S}], I_E[\mathcal{S}] \leftarrow \Phi(H, \mathcal{S}, t_{\mathcal{V}}, t_E)$
    if $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$ **then**
        return $\mathcal{S}$
    $E' \leftarrow E \setminus I_E[\mathcal{S}]$
    $[H]_2 \leftarrow$ TwoSection$(H(\mathcal{V}, E'))$
return $\mathcal{S}$

8.3.2 A Subtractive Heuristic

The heuristic described in this section extends the TSS algorithm proposed by Cordasco et al. in [58] for graphs. At each iteration, the heuristic greedily prunes vertices or hyperedges from the input hypergraph H unless a specific condition occurs, making a vertex being added to the seed set $\mathcal{S}$. The algorithm stops when all vertices have either been discarded or selected as seeds. The pruning phase is designed based on a rule that tries to balance the capability of a vertex (hyperedge) to influence other hyperedges (vertices) and its *easiness* (or *hardness*) to be influenced by other hyperedges (vertices). We denote this heuristic as subtractive as, in contrast to the additive procedures, it first prunes the hypergraph and then adds a vertex to the seed set if and only if its neighbors cannot influence that vertex.

SubTSSH

Algorithm 4 details the pseudo-code of SubTSSH. This procedure is made up by three main parts, encoding a particular condition of the influence status of vertices and hyperedges.

- *Case 1.* This case captures whether a vertex (Case 1.a) or a hyperedge (Case 1.b) has a threshold of 0. If this happens, the element can be pruned from the hypergraph as it can be considered self-influenced.
- *Case 2.* This phase identifies whether a vertex v has a residual degree lower than its residual threshold, i.e., $\deg(v) < t_{\mathcal{V}}(v)$. In this case, the vertex cannot be activated by its neighbors; thus it must be added to the seed set.
- *Case 3.* In this last case, the algorithm selects either a vertex (Case 3.a) or a hyperedge (Case 3.b) to prune from the hypergraph. In this implementation, SubTSSH picks a vertex and a hyperedge with the lowest residual threshold-degree ratio and then removes the element easier to influence.

Algorithm 4 SubTSSH$(H = (\mathcal{V}, E), t_{\mathcal{V}}, t_E)$

$\mathcal{S} \leftarrow \emptyset$
while $\mathcal{V} \neq \emptyset$ **do**
  if $\exists\, u \in \mathcal{V} \mid t_{\mathcal{V}}(u) = 0$ **then** ▷ Case 1.a: if a vertex exists with threshold 0, it is self-influenced
    $\mathcal{V} \leftarrow \mathcal{V} \setminus \{u\}$
    UpdateThresholds(H, u) ▷ Reduce the thresholds of the edges containing u
    UpdateSizes(H, u) ▷ Reduce the size of the edges containing u
  else
    if $\exists\, e \in E \mid t_E(e) = 0$ **then** ▷ Case 1.b: if an edge exists with threshold 0, it is self-influenced
      $E \leftarrow E \setminus \{e\}$
      UpdateThresholds(H, e) ▷ $\forall v \in e$, reduce the threshold of v
      UpdateDegrees(H, e) ▷ Reduce the degree of the vertices belonging to e
    else
      if $\exists\, u \in \mathcal{U} \mid deg(u) < t_{\mathcal{V}}(u)$ **then** ▷ Case 2: v cannot be influenced by its neighbors
        $S \leftarrow S \cup \{u\}$;
        $\mathcal{V} \leftarrow \mathcal{V} \setminus \{u\}$
        UpdateThresholds(H, u)
        UpdateSizes(H, u)
      else ▷ Remove a vertex v or an edge e
        $u \leftarrow \arg\min_{v \in \mathcal{V}} \frac{t_{\mathcal{V}}(v)}{\deg(v)(\deg(v)+1)}$
        $e \leftarrow \arg\min_{e' \in E} \frac{t_E(e')}{\kappa(e')(\kappa(e')+1)}$
        if $\frac{t_{\mathcal{V}}(u)}{\deg(u)(\deg(u)+1)} < \frac{t_E(e)}{\kappa(e)(\kappa(e)+1)}$ **then** ▷ Case 3.a: Remove u
          $\mathcal{V} \leftarrow \mathcal{V} \setminus \{u\}$
          UpdateSizes(H, u)
        else ▷ Case 3.b: Remove e
          $E = E \setminus \{e\}$
          UpdateDegrees(H, e)
return $\mathcal{S}$

> The particular choice of the denominators derives from a theoretical analysis performed by Cordasco et al. for the TSS algorithm on graphs [58].

SubTSSH proceeds as follows. As long as the hypergraph is not empty and no special conditions occur (e.g., Cases 1 or 2), a vertex u (Case 3.a) or an hyperedge e (Case 3.b) is selected according to a given function (see Case 3) and it is pruned from the hypergraph. When a vertex (hyperedge) is removed, its incident hyperedges (vertices) update their size (degree) as they cannot count on u (e) anymore to become influenced. Due to this update, some vertices in the residual hypergraph may remain with less *usable* hyperedges (if a vertex $u \in \mathcal{V}$ has $\deg(u) < t_{\mathcal{V}}(u)$). In such a case (see Case 2), these vertices are added to the seed set $\mathcal{S}$ and removed from H. Then, their incident hyperedges' size and thresholds are updated (i.e., decreased by 1) since they have one more influenced neighbor. In the last case (see Case 1), the residual hypergraph contains a vertex u (hyperedge e) whose threshold decreased to 0. This situation means that the vertices already added to the seed set $\mathcal{S}$ are enough to influence u (e). At this point, u (e) is removed from H, and both the size and the thresholds of its incident hyperedges (vertices) are updated (i.e., decreased by 1) since they obtained one more influenced neighbor.

8.3.3 An Optimization Procedure

Being greedy-based approaches, all the described algorithms ensure that $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$, but they do not guarantee that the cardinality of $\mathcal{S}$ is the optimal solution. Further, we experimentally observed that the evaluated solutions often contained unnecessary seed vertices. For these reasons, we developed a cleaning procedure called $OptimizationStrategy$ and listed in Algorithm 5, which iteratively removes all redundant vertices from a solution obtained by the above algorithms.

Definition 8.4 : Unnecessary Vertex

Given a seed set $\mathcal{S}$ s.t. $I_{\mathcal{V}}[\mathcal{S}] = \mathcal{V}$, a vertex $v \in \mathcal{S}$ is *unnecessary* if $I_{\mathcal{V}}[\mathcal{S} \setminus \{v\}] = \mathcal{V}$.

Algorithm 5 OptimizationStrategy$(H = (\mathcal{V}, E), t_{\mathcal{V}}, t_E, \mathcal{S})$

1: Let $[s_1, s_2, \ldots, s_t]$ be a sequence of all vertices in $\mathcal{S}$ in non-increasing order of their degree.
2: $\mathcal{S}_\pi \leftarrow \mathcal{S}$
3: **for** $i \in [1, t]$ **do**
4: $\quad I_{\mathcal{V}}[\mathcal{S}_\pi], I_E[\mathcal{S}_\pi] \leftarrow \Phi(H, \mathcal{S}_\pi \setminus \{s_i\}, t_{\mathcal{V}}, t_E)$
5: $\quad$ **if** $I_{\mathcal{V}}[\mathcal{S}_\pi] = \mathcal{V}$ **then**
6: $\quad\quad \mathcal{S}_\pi \leftarrow \mathcal{S}_\pi \setminus \{s_i\}$
7: return $\mathcal{S}_\pi$

8.4 State-of-the-art Comparison

In our work, we address the TSS problem in social networks abstracted as hypergraphs. To study spreading phenomena over such structures, we extended the classical LT diffusion model to capture the high-order interactions characterizing the underlying network and resembling that a person may be influenced by their groups and vice-versa.

The proposed LT high-order diffusion model embodies the first main difference of our study with respect to other literature on the topic. Specifically, our model spreads influence iteratively from vertices to hyperedges and from hyperedges to vertices (see Section 8.2.1). In this model, hyperedges play an active role in the diffusion process as they can be influenced (if the number of influenced vertices within the hyperedge is at least equal to the associated threshold) and can spread influence once activated.

Only two works among the articles analyzed in Chapter 4 embed a similar idea within the diffusion process. Gangal et al. [78] extended the

IC model to hypergraphs, considering a hyperedge activated if most of its vertices are influenced at the end of the diffusion process. However, the primary objective of this work was to maximize the number of hyperedges influenced rather than the number of vertices influenced. Further, they used hyperedges as influence carriers, allowing the influence to flow from one vertex to another through them. Later on, Zhu et al. [198] analyzed a similar problem with the goal of maximizing the expected number of influenced groups. In this case, Zhu et al. considered a group influenced if at least a fixed percentage β of users in it are influenced under the classical IC model. According to their model, hyperedges can be influenced but are not actively involved in the diffusion process. Other works consider hyperedges as a tool to model different types of influence (for instance, hyperedges represent either direct or activity-based influence [185]) or to identify groups of vertices a vertex can spread influence to (such as in [134, 173]). Further, all works discussed in Chapter 4 except Wang et al. [185] analyze the problem under the IC setting.

Another difference lies in the hypergraph model used. In our work, we represent the underlying network with an undirected hypergraph. Similarly, even though in a different problem setting, Borgs et al. [44], Gangal et al. [78], Ma et al. [134], and Suo et al. [173] use the same hyper-network model. Conversely, Zhu et al. abstract the network with directed hypergraphs in all three works reported in this dissertation [198, 200, 199].

The last primary difference is the specific problem addressed. We specifically study the TSS problem generalized to hypergraphs. In contrast, the other studies deal with the classical SIM problem on hypergraphs ([200, 199, 185]), its variant in which the objective is maximizing the number of eventually influenced groups/hyperedges ([78, 198]) or exploring information transmission patterns via high-order relations ([134, 173]).

The work of Borgs et al. [44] represents an exception to this pattern as the SIM problem the authors deal with is defined on directed graphs, and hypergraphs are used as a representation of an influence estimation. In this work, a hyperedge represents all vertices influenced by a single

vertex under the IC model; hence, we do not have the notion of hyperedge as a group. The seed set is then selected on the so-obtained hypergraph exploiting a greedy procedure, which works similarly to DynamicGreedy (see Section 8.3).

8.5 Experiment Setting

To evaluate the effectiveness of the greedy-based heuristics discussed in Section 8.3, we performed a bunch of experimental scenarios varying the activation threshold for both vertices and hyperedges. Specifically, we assessed their performance on several random and real-world networks, considering both the cardinality of the solution provided (the smaller, the better) and their execution times. This section details the data used in our experiments and the designed experimental scenarios, divided according to the network nature. Section 8.6 presents and discusses the outcomes obtained.

Code Availability

All heuristics and experiments are implemented in Julia, exploiting the library `SimpleHypergraphs.jl` (see Chapter 7). The code is open-source and available on the following GitHub public repository `https://github.com/pszufe/LTMSim.jl` and on Zenodo [21].

8.5.1 Random Networks

In this part, we provide a description of the random hypergraphs used in the first set of evaluation scenarios.

Data sets

As benchmark hypergraphs, we used random generated hypergraphs with and without structural constraints. Specifically, we employed: (i) random hypergraphs with no structural properties, (ii) hypergraphs generated via the preferential-attachment rule, (iii) $k-$uniform and (iv) $d-$regular hypergraphs. We refer the reader to Section 7.3.2 for a description of the generative models offered by SimpleHypergraphs.jl.

Evaluation Scenarios

We performed three experimental scenarios. In the first and second scenarios, we fixed each vertex threshold to a random value between 1 and its degree. In the last scenario, each vertex threshold varies proportionally - from 0.1 to 0.9 - to the degree of the vertex. In all experiments, we set each hyperedge activation threshold proportional to its degree scaled of factor 0.5 (majority policy).

Regarding the networks used, in the first scenario, we run the heuristics on random hypergraphs with no structural properties and hypergraphs generated via the preferential-attachment rule. We ranged the hypergraph size, using $\{100, 200, 400, 800\}$ vertices and hyperedges. In the second scenario, we experimented the heuristics on $k-$ uniform and $d-$ regular random hypergraphs of size $(n = 500, m = 500)$, ranging the value of k and d in $\{10, 20, 40, 80\}$. In the third and last scenario, we generated a random hypergraph with 500 vertices and 500 hyperedges using all four generative models, fixing $k = 80$ and $d = 80$, for the $k-$ uniform and $d-$ regular random hypergraphs, respectively. We simulated the execution of each heuristic 50 times.

Experimental design

The choice of the parameters to generate random hypergraphs has been made to assure a reasonable computational time when run-

ning all heuristics. The only structural constraint applies to the values of k and d: we set both k and $d \ll |\mathcal{V}|$ not to have any hyperedge containing all vertices and any vertex linked to all other vertices.

Table 8.1: Threshold settings in the experimental scenarios when using random hypergraphs.

	Vertex thresholds		Random Model			
	Random	**Proportional**	**No constraints**	**Preferential attachment**	**$k-$ uniform**	**$d-$ regular**
Scenario 1	✓		✓	✓		
Scenario 2	✓				✓	✓
Scenario 3		✓	✓	✓	✓	✓

8.5.2 Real-world Networks

In this part, we describe the real-world hypergraphs used in the second set of evaluation scenarios.

Data sets

As benchmark hypergraphs, we used 11 networks generated by real-world data sets, downloaded from the ARB [35], Mendeley [56] and GitHub [103] repositories. A more detailed description of each data set is provided in Chapter 6. Table 8.2 and Table 8.3 summarize the dimension of each hypergraph, along with the number of edges in the corresponding clique-expansion. The size shown refers to the final dimension of the hypergraphs after having removed 0-degree vertices and empty hyperedges.

Table 8.2: Benchmark hypergraphs details (1/2).

	Algebra	Amazon	DBLP	Email-Enron	Email-W3C	Geometry
$\lvert\mathcal{V}\rvert$	423	4,989	2,727	2,807	5,601	580
$\lvert E\rvert$	1,268	1,176	874	5,000	6,000	1,193
$\lvert E\rvert_{[H_2]}$	50,209	11,590	4,298	88,926	11,130	205,127
Source	[35]	[35]	[56]	[35]	[35]	[35]

Table 8.3: Benchmark hypergraphs details (2/2).

	GoT	Music-Rev	NBA	Restaurants-Rev	Bars-Rev
$\lvert\mathcal{V}\rvert$	577	1,106	567	565	1,234
$\lvert E\rvert$	3,840	694	2,163	601	1,194
$\lvert E\rvert_{[H_2]}$	23,083	149,288	456,251	31,241	167,000
Source	[103]	[35]	[56]	[35]	[35]

Data set availability

All hypergraphs used in this experimental part are available at the following public GitHub repository `https://github.com/pszufe/LTMSim.jl` and on Zenodo [20].

Evaluation Scenarios

We performed four experimental scenarios. In the first and second scenarios, we fixed each vertex threshold to a random value between 1 and its degree. In the other two scenarios, each vertex threshold varies proportionally - from 0.2 to 0.8 - to the degree of the vertex. Concerning the

hyperedge thresholds, we fixed each edge threshold to a random value between 1 and its degree in the first and third scenarios. In the remaining settings, we set each hyperedge activation threshold proportional to its degree scaled of factor 0.5 (majority policy). Table 8.4 summarizes the threshold settings in the four experimental scenarios. We simulated the execution of each heuristic followed by the optimization procedure 50 times per experiment. We also repeated the experiment simulating the execution of each heuristic without applying the optimization step.

Table 8.4: Threshold settings in the experimental scenarios when using real-world hypergraphs.

	Vertex thresholds		Hyperedge thresholds	
	Random	**Proportional**	**Random**	**Majority policy**
Scenario 1	✓		✓	
Scenario 2	✓			✓
Scenario 3		✓	✓	
Scenario 4		✓		✓

8.6 Results

This section presents and discusses the results of the experimental scenarios introduced in Section 8.5. Specifically, we first focus on the outcomes obtained over random and then over real-world hyper-networks. Finally, we examine the time performance of the proposed heuristics.

8.6.1 Random Networks

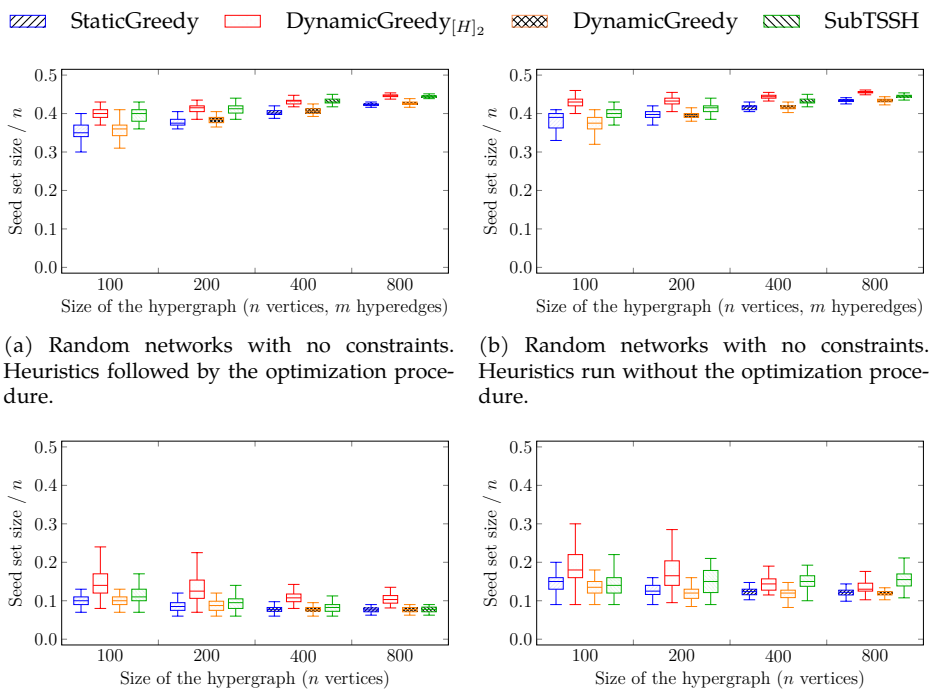
Here, we describe the results related to the experimental scenarios run over random hyper-networks and introduced in Section 8.5.1.

Scenario 1 - Random thresholds, Models: no constraints, preferential attachment

Figure 8.3 shows the results obtained on random hypergraphs, generated without structural constraints (see Figures 8.3a and 8.3b) and with the preferential-attachment rule (see Figures 8.3c and 8.3d), varying the vertex degree and the hyperedge size. We fixed each vertex threshold to a random value between 1 and the vertex degree, varying it at each experiment run, and a majority policy for hyperedges. The seed set size is reported as the ratio over the total number of vertices to enable comparison across all hyper-networks.

Looking at the results related to hypergraphs with no constraints, we can note three major points. First, no heuristic outperforms the others, even though DynamicGreedy tends to obtain slightly smaller seed sets when the optimization procedure is not applied. The greater the size of the network is, the smaller this difference becomes (as well as the variance of the results). Second, the optimization procedure can reduce the seed set size up to 3% on average. Hence, we cannot observe a drastic improvement of the heuristics. Third, all heuristics scale pretty well to the network size increase, always reaching a seed set size near 40% of all vertices.

We can notice a similar behavior when using the hyper-networks generated with the preferential-attachment rule. Also in this case, DynamicGreedy reaches moderately better results when the optimization procedure is not applied. However, even though we cannot observe significant variations when the optimization procedure is applied, we can see that DynamicGreedy$_{[H]_2}$ tends to have the highest variance in the results and, generally, obtains the worst outcomes. In contrast to the previous results, the optimization procedure helps reduce the seed set size from 3% up to 12% on average. This bigger improvement suggests that this kind of network is more difficult to influence, and it is more likely for the heuristics to select superfluous vertices. As before, all heuristics scale well to the network size increase. This result was somewhat expected as all ad-



(a) Random networks with no constraints. Heuristics followed by the optimization procedure.

(b) Random networks with no constraints. Heuristics run without the optimization procedure.

(c) Preferential attachment networks. Heuristics followed by the optimization procedure.

(d) Preferential attachment networks. Heuristics run without the optimization procedure.

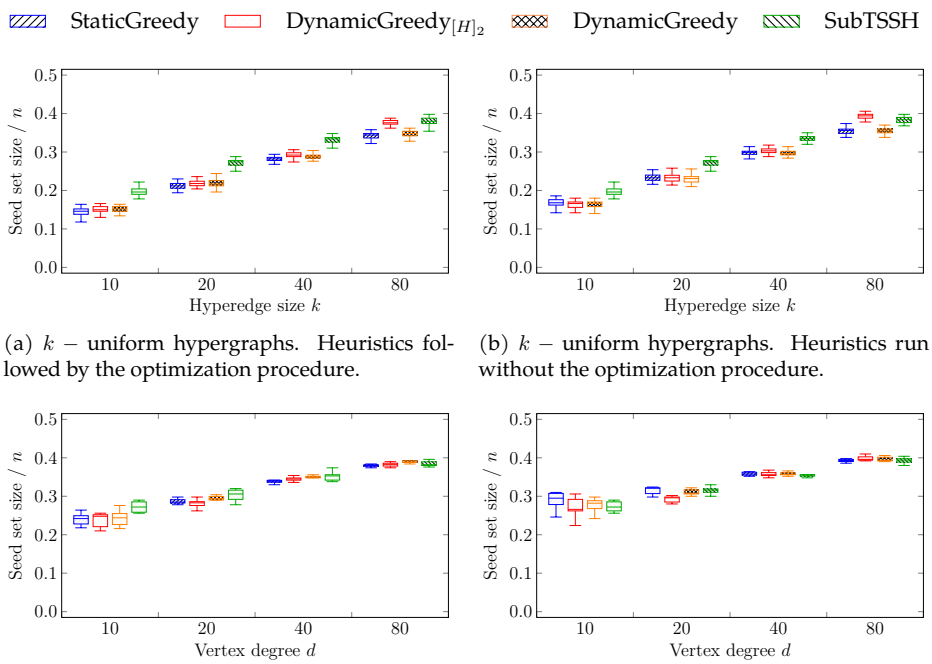
Figure 8.3: Scenario 1 — Experiments on random hypergraphs, generated without structural constraints (see Figures 8.3a and 8.3b) and with the preferential-attachment rule (see Figures 8.3c and 8.3d), varying the vertex degree and the hyperedge size. Each vertex threshold is fixed to a random value between 1 and the vertex degree. A fixed threshold of 0.5 is used for hyperedges.

ditive heuristics select the vertices with a higher degree (hence, the network hubs), known to be a small fraction of all available vertices. On the contrary, the optimization procedure has little or no effects on SubTSSH. This procedure selects vertices with the best balance between the influence potential and the easiness to become active; thus, it tends to insert the vertices with both higher degree and threshold in the seed set. We can empirically observe that this translates into selecting less unnecessary vertices.

Scenario 2 - Random thresholds, Models: $k-$ uniform, $d-$ regular

In a second experiment, we exploited $k-$ uniform and $d-$ regular hypergraphs, varying the hyperedge size k and the vertex degree d. We used the same threshold setting of Scenario 1, fixing each vertex threshold to a random value between 1 and the vertex degree, and a majority policy for hyperedges. Figure 8.4 shows the distribution of the ratio of the seed set size over the total number of vertices n per hypergraph obtained by running the four heuristics with and without the optimization step.

Observing the figure, we can note a pattern similar to the one discussed in Scenario 1. Also in this experiment, no heuristic clearly outperforms the others. Focusing our attention on $k-$ uniform hypergraphs (see Figures 8.4a and 8.4b), all additive heuristics achieve very similar results, while SubTSSH tends to choose a bigger seed set. We can somewhat observe the opposite tendency on $d-$ regular hypergraphs when the optimization procedure is not used (see Figure 8.4d). Further, we can observe a higher variance for small values of the parameter d. As in Scenario 1, the optimization procedure reduces up to 3% and 8% the seed set size in the case of $k-$ uniform and $d-$ regular hypergraphs, respectively. Again, this procedure has a deeper impact on the solution found by additive heuristics rather than SubTSSH as clearly evident in the case of $d-$ regular networks (see Figures 8.4c and 8.4d). The last interesting thing to note is that, in general, $k-$ uniform hypergraphs require a target set of smaller size compared to $d-$ regular hypergraphs. This outcome directly derives from the inherent structure of the generated networks. In particular, all vertices have the same degree d in a $d-$regular hypergraphs. This means that all vertices are treated equally by the additive heuristics, which prefer higher degree vertices. We can make a similar statement for SubTSSH, which prunes the easiest to influence vertices based on their degree and thresholds from the hypergraphs. Being all vertex degrees equal, all heuristics cannot count on this information to choose the best seed set.



(a) k − uniform hypergraphs. Heuristics followed by the optimization procedure.

(b) k − uniform hypergraphs. Heuristics run without the optimization procedure.

(c) d−regular hypergraphs. Heuristics followed by the optimization procedure.

(d) d − regular hypergraphs. Heuristics run without the optimization procedure.

Figure 8.4: Scenario 2 — Experiments on k − uniform (see Figures 8.4a and 8.4b) and d − regular (see Figures 8.4c and 8.4d) hypergraphs, varying the hyperedge size k and the vertex degree d. Each vertex threshold is fixed to a random value between 1 and the vertex degree. A fixed threshold of 0.5 is used for hyperedges.

Scenario 3 - Proportional vertex thresholds, All models

In this last scenario, we repeated the experiments performed in Scenario 1 and Scenario 2 but ranging each vertex activation threshold proportionally to its degree - from 0.1 to 0.9, while we still used a majority policy for hyperedges. In this case, we generated random hypergraphs of fixed size $n = m = 500$, setting $k = d = 80$ to create of k − uniform and d − regular hypergraphs. Figure 8.5 depicts the results obtained for each hypergraph type. We have to remind the reader that the higher the threshold, the

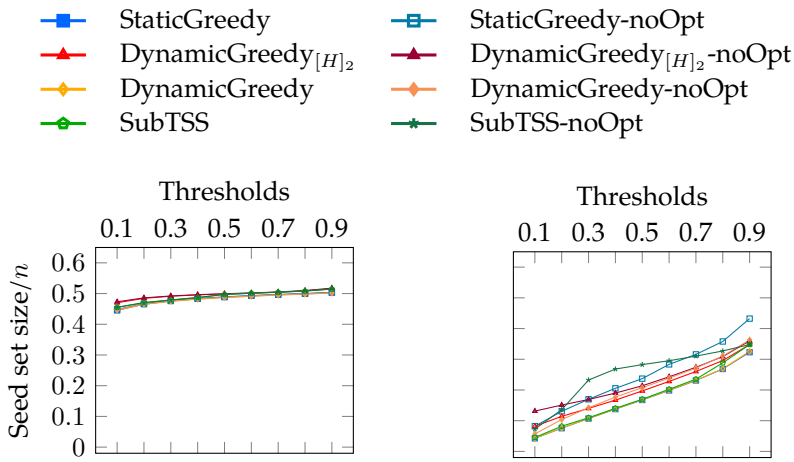
harder it is activating the vertex.

The first interesting thing to note is that all heuristics practically achieve the same performance in the case of random hypergraphs with no structural constraints (see Figure 8.5a) and $d-$ regular hypergraphs (see Figure 8.5d). This result is in line with the previous scenarios: in both cases, there is not enough information conveyed by the vertex degrees to allow the heuristics to choose (and differentiate) a better seed set. Further, there is no difference between the seed set size achieved by each heuristic and its variant followed by the optimization procedure.

We can observe a significantly different picture in Figure 8.5b, showing the results relative to random hypergraphs generated with the preferential attachment rule. In this case, we can note three major points. First, as we can also observe in Figure 8.3c in Scenario 1, the optimization procedure is able to reduce the original seed set size evaluated by all four heuristics. Second, DynamicGreedy$_{[H]_2}$ followed by the optimization step achieves the worst results compared to the other heuristics optimized. Third, the worst performance is not reached consistently by one specific heuristic, but it varies with the vertex thresholds. For instance, when the thresholds are small, the performance of DynamicGreedy$_{[H]_2}$ is poor, but for larger values, its performance improves and becomes very close to the DynamicGreedy (without optimization). SubTSSH with no optimization evaluates the worst seed set for thresholds from 0.3 to 0.6 to then improve and finally reach the same seed set size of SubTSSH followed by the optimization step. StaticGreedy without optimization tends to achieve the worst seed set size, reaching the higher values when the threshold is equal to 0.8 or 0.9.

Figure 8.5c shows results kind of in the middle of the two previously discussed. As for random hypergraphs with no constraints and $d-$regular hypergraphs, the optimization procedure does not improve the initial seed set size. Once again, this result was expected given the outcomes described in Scenario 2 (see Figure 8.4). However, we can observe that DynamicGreedy$_{[H]_2}$ (with and without optimization procedure) systematically achieves the worst performance. Similarly, SubTSSH obtains slightly

worst results compared to StaticGreedy and DynamicGreedy, but they decrease to the increase of the vertex thresholds reaching the same outcomes of DynamicGreedy$_{[H]_2}$ at the end.



(a) Random hypergraphs with no constraints.

(b) Preferential attachment hypergraphs.

(c) $k-$ uniform hypergraphs.

(d) $d-$ regular hypergraphs.

Figure 8.5: Scenario 3 — Experiments on random hypergraphs $H = (\mathcal{V}, E)$ of size $n = m = 500$, built using all generative models. Each vertex threshold varies proportionally - from 0.1 to 0.9 - to the degree of the vertex. A fixed threshold of 0.5 is used for hyperedges. The value of k and d for the k-uniform and d-regular hypergraphs is set to 80.

8.6.2 Real-world Networks

In this section, we describe the results related to the experimental scenarios run over real-world hyper-networks and introduced in Section 8.5.2.

Scenario 1 - Random Thresholds

In this first experimental scenario, we fixed each vertex (hyperedge) threshold to a random value between 1 and its degree (size), varying it at each experiment run. Figure 8.6 shows the distribution of the seed set size per hypergraph using the four heuristics described in Section 8.3, followed by the optimization procedure. Figure 8.7 reports the same information zooming into each data set comparing the four heuristics with (left—orange) and without (right—green) the optimization procedure. The size of the seed set is reported as the ratio over the total number of vertices to enable comparison across all data sets.

The first interesting aspect to note is that the optimization phase has a meaningful impact on all three additive heuristics - StaticGreedy, DynamicGreedy$_{[H]_2}$, and DynamicGreedy. This behavior is highly emphasized when looking at the StaticGreedy procedure, where the algorithm selects almost all of the vertices when run without optimization. As expected,

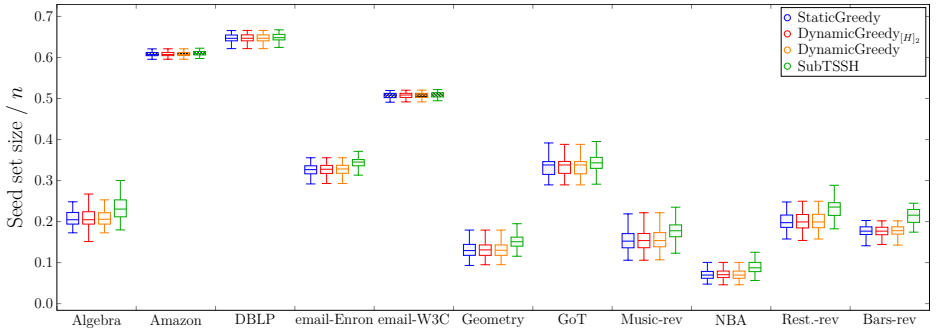
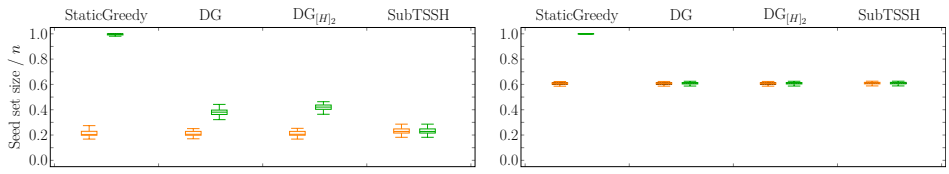
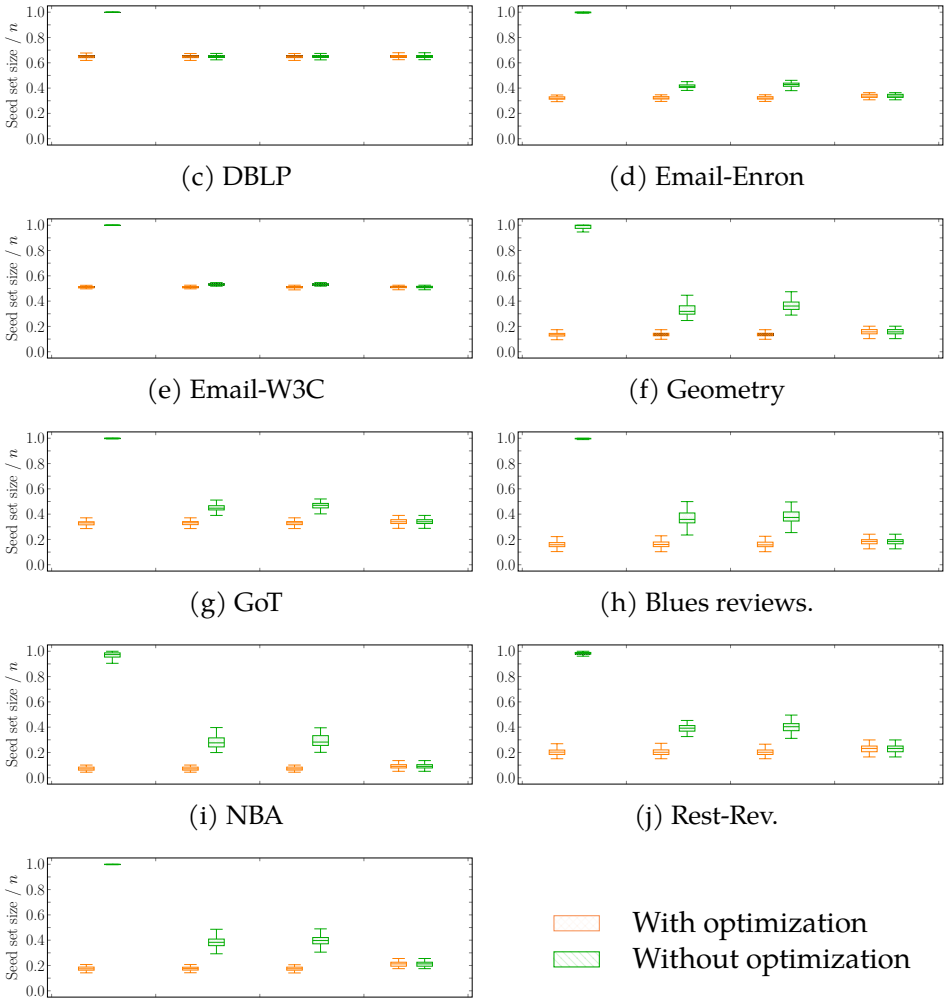


Figure 8.6: Scenario 1 — Ratio of the seed set size over the total number of vertices n per hypergraph, fixing each vertex (hyperedge) threshold to a random value between 1 and its degree (size).

this outcome resembles the results obtained when running the heuristics on hypergraphs generated with the preferential attachment rule (see Scenario 1 and Scenario 3 of Section 8.6.1), as real-world hyper-networks have a heavy-tail hyperedge distribution [123]. The rationale behind why the optimization procedure has a more significant impact on the solution found by additive heuristics has to be sought in how these work. StaticGreedy, DynamicGreedy$_{[H]_2}$, and DynamicGreedy only consider the (residual) vertex degree when selecting vertices to add to the seed set. However, there is no guarantee that higher-degree vertices constitute a better seed set since vertex and hyperedge thresholds are randomly assigned. Second, the cleaning procedure carried out after selecting the initial seed set leads to achieving very similar results when using the additive heuristics. This outcome may happen because the three heuristics choose a very similar set of core vertices (i.e., the vertices remaining after the optimization phase), even though they initially select different seed sets. On the other hand, the optimization phase has a very low, if not zero, impact on reducing the initial seed set found by the subtractive heuristic SubTSSH (since it considers both degree and vertex thresholds). Thus, running the algorithm with and without optimization brings consistent results, always comparable with the outcomes of the additive heuristics. As a result, when using the SubTSSH algorithm, the optimization phase may also be skipped by paying a minimal price on the dimension of the final seed set.



(a) Algebra (b) Amazon



(c) DBLP (d) Email-Enron

(e) Email-W3C (f) Geometry

(g) GoT (h) Blues reviews.

(i) NBA (j) Rest-Rev.

(k) Bars Reviews.

Figure 8.7: Scenario 1 — Ratio of the seed set size over the total number of vertices n per hypergraph, fixing each vertex (hyperedge) threshold to a random value between 1 and its degree (size). The acronyms DG and $\text{DG}_{[H]_2}$ stand for DynamicGreedy and $\text{DynamicGreedy}_{[H]_2}$, respectively. Each plot compares the seed set size obtained by running each heuristic with (left, orange) and without (right, green) optimization procedure.

Scenario 2 - Random Vertex Thresholds, Majority Policy on Hyperedges

In a second experiment, we fixed each vertex threshold to a random value between 1 and its degree, varying it at each experiment run. We used a majority policy for the hyperedge thresholds instead. Figure 8.8 shows the distribution of the ratio of the seed set size over the total number of vertices n per hypergraph obtained by running the four heuristics with the optimization step. Figure 8.9 reports the same information zooming into each data set comparing the four heuristics with (left, orange) and without (right, green) the optimization procedure. The results of this experiment resemble the outcomes of Scenario 1. As in the previous case, additive heuristics (plus the optimization procedure) perform slightly better than SubTSSH in obtaining the smallest seed set.

Among all data sets, when it comes to selecting the seed set for the hypergraphs associated with *Amazon* and *DBLP*, all heuristics (with and without optimization procedure) pick at least half of the total number of vertices. This result is emphasized by the smaller number of vertices (in percentage) required to fully influence the other networks. A similar pattern also happens in Scenario 1 (see Figure 8.6). The reason for this

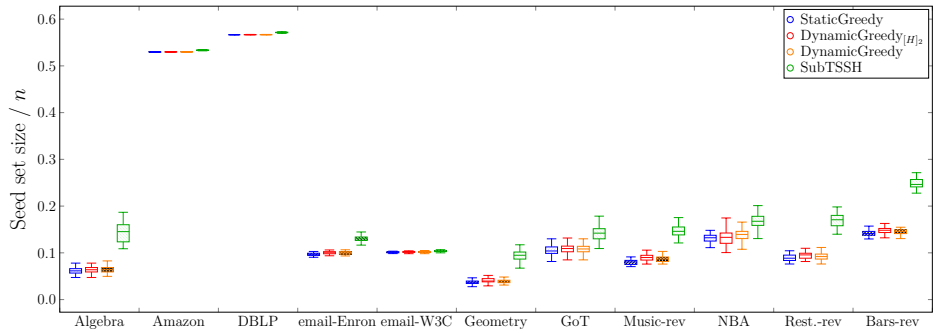
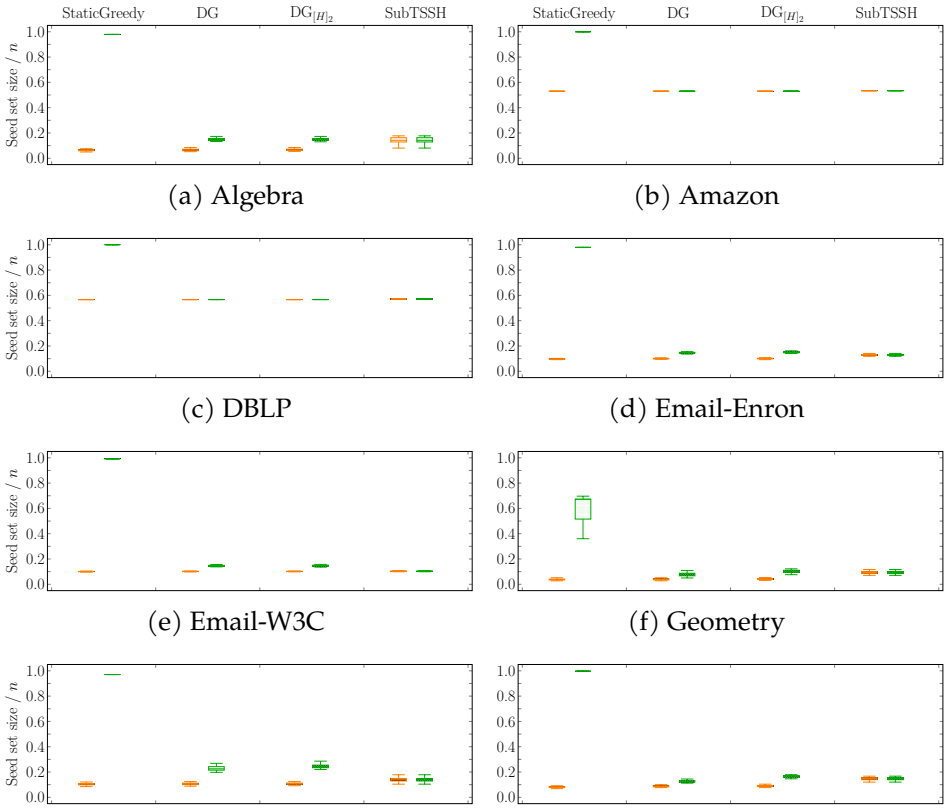


Figure 8.8: `Scenario 2` - Ratio of the seed set size over the total number of vertices n per hypergraph, fixing each vertex threshold to a random value between 1 and its degree and using a majority policy for hyperedge thresholds.

behavior has to be sought in the topology of the networks. What differentiates the *Amazon* and *DBLP* hypergraphs from the others is that their number of vertices is at least 68% higher than their number of hyperedges. In addition, the majority of the vertices of the two networks has degree equal to 1 (97.89% for *Amazon*, 97.21% for *DBLP*). Being involved in only one relation, these vertices are difficult to influence and, at the same time, cannot help in influencing other vertices; thus, confirming how the topology of the network strongly determines the minimum target set needed to influence the whole hypergraph. As expected, StaticGreedy performs the worst when not followed by the cleaning step, selecting in all hyper-

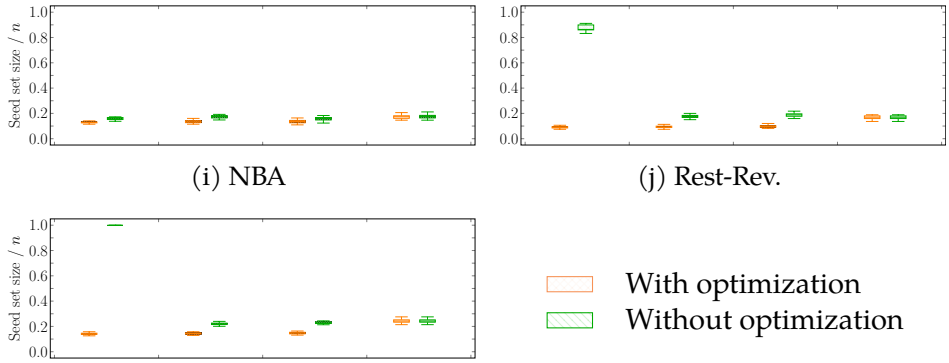


(a) Algebra (b) Amazon

(c) DBLP (d) Email-Enron

(e) Email-W3C (f) Geometry

(g) GoT (h) Blues reviews.



(i) NBA

(j) Rest-Rev.

(k) Bars Reviews.

Figure 8.9: Scenario 2 — Ratio of the seed set size over the total number of nodes n per hypergraph, fixing each node threshold to a random value between 1 and its degree and using a majority policy for hyperedge thresholds. The acronyms DG and $\text{DG}_{[H]_2}$ stands for DynamicGreedy and $\text{DynamicGreedy}_{[H]_2}$, respectively.

graphs but *NBA* the highest number of vertices (if not almost all vertices). In line with the experiments on random networks, this outcome suggests that using a dynamic approach is helpful to select a reasonable seed set.

Scenario 3 - Proportional node thresholds, random hyperedge thresholds.

In this third experimental scenario, we varied each node threshold proportionally - from 0.2 to 0.8 - to the degree of the node and fixed each hyperedge threshold to a random value between 1 and its size. It is worth stressing that the higher the threshold, the harder it is to influence the given node or hyperedge.

Figure 8.10 shows the ratio of the seed set size over the total number of nodes n separately for each hypergraph. These outcomes look like the ones previously discussed in Scenario 1 and Scenario 2. Overall, the additive heuristics followed by the optimization procedure performed better

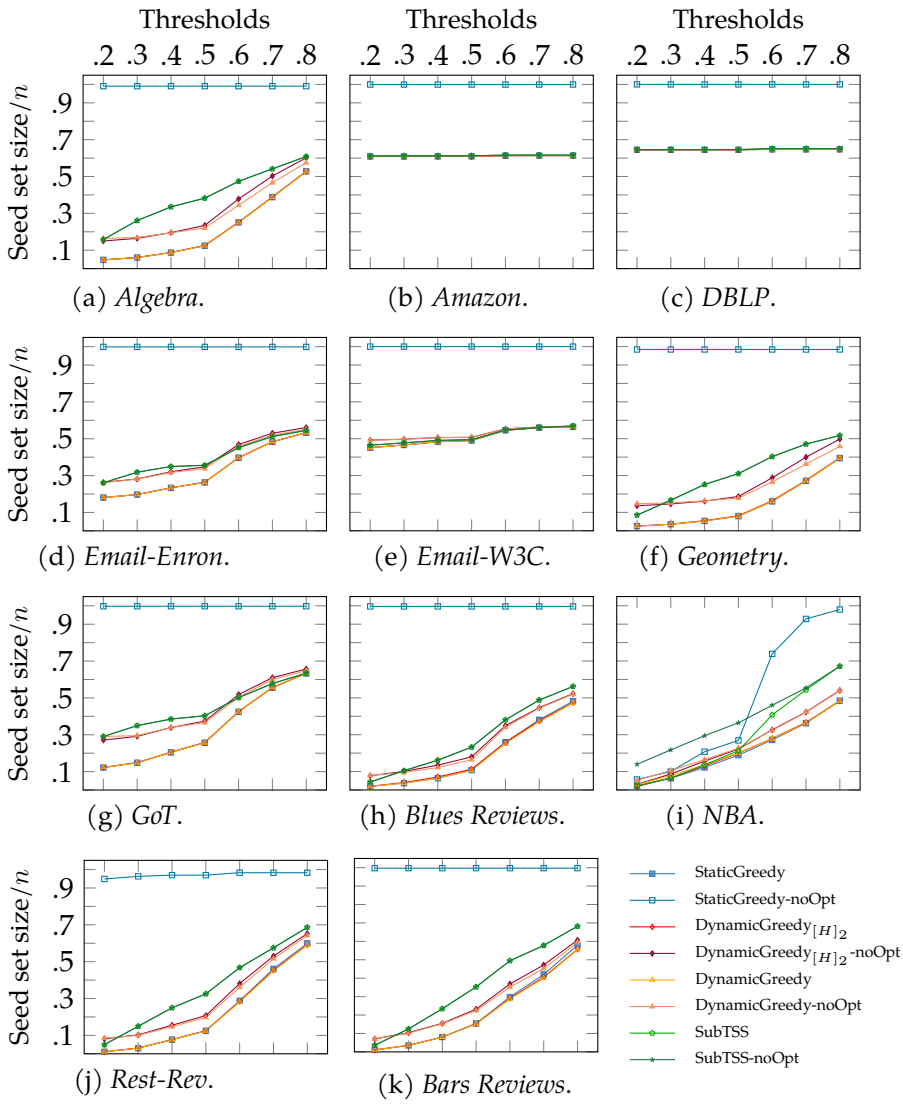


Figure 8.10: Scenario 3 - Ratio of the seed set size over the total number of nodes n per hypergraph, varying each node threshold proportionally - from 0.2 to 0.8 - to its degree, and fixing each hyperedge threshold to a random value between 1 and its size.

in all data sets, consistently achieving a smaller seed set than SubTSSH, regardless of the node threshold value. Generally, using proportional node thresholds, DynamicGreedy and DynamicGreedy$_{[H]_2}$ without the cleaning procedure tend to perform equal or better than SubTSSH (with and without optimization). A plausible explanation for this behavior must be sought in how each algorithm selects the next node to add to the seed set. Specifically, DynamicGreedy and DynamicGreedy$_{[H]_2}$ choose the nodes to put in the seed set only based on their degree. In this specific setting, selecting the nodes with a higher degree increases the probability of influencing a higher number of nodes (being each node's threshold proportional to its degree). On the other hand, SubTSSH prunes from the hypergraph nodes that are easy to influence, even though they may have many neighbors; hence, removing possible good candidates. As already discussed in the previous two scenarios, the StaticGreedy algorithm without the cleaning step always produces a seed containing almost all nodes. Further, as before, optimizing the result of the SubTSSH procedure brings little or no improvements to the seed set size.

Again, this experiment draws attention to the *Amazon* and *DBLP* hypergraphs, as it clearly stands out that all heuristics have the same performance (as also happens in the first two experiments, see Figures 8.6 and 8.8). We can further observe the same behavior for the *Email-W3C* hypergraph. As already discussed, the reason for that lies in the number of nodes being part of only one relation (80.57% of the nodes have degree 1).

Scenario 4 - Proportional node thresholds, majority policy on hyperedges.

In the fourth and last experimental scenario, we varied each vertex threshold proportionally - from 0.2 to 0.8 - to the degree of the vertex, while we used a majority policy for hyperedges.

Figure 8.11 shows the ratio of the seed set size over the total number of vertices n separately for each hypergraph. We can observe a familiar picture, where the three additive heuristics - followed by the optimization

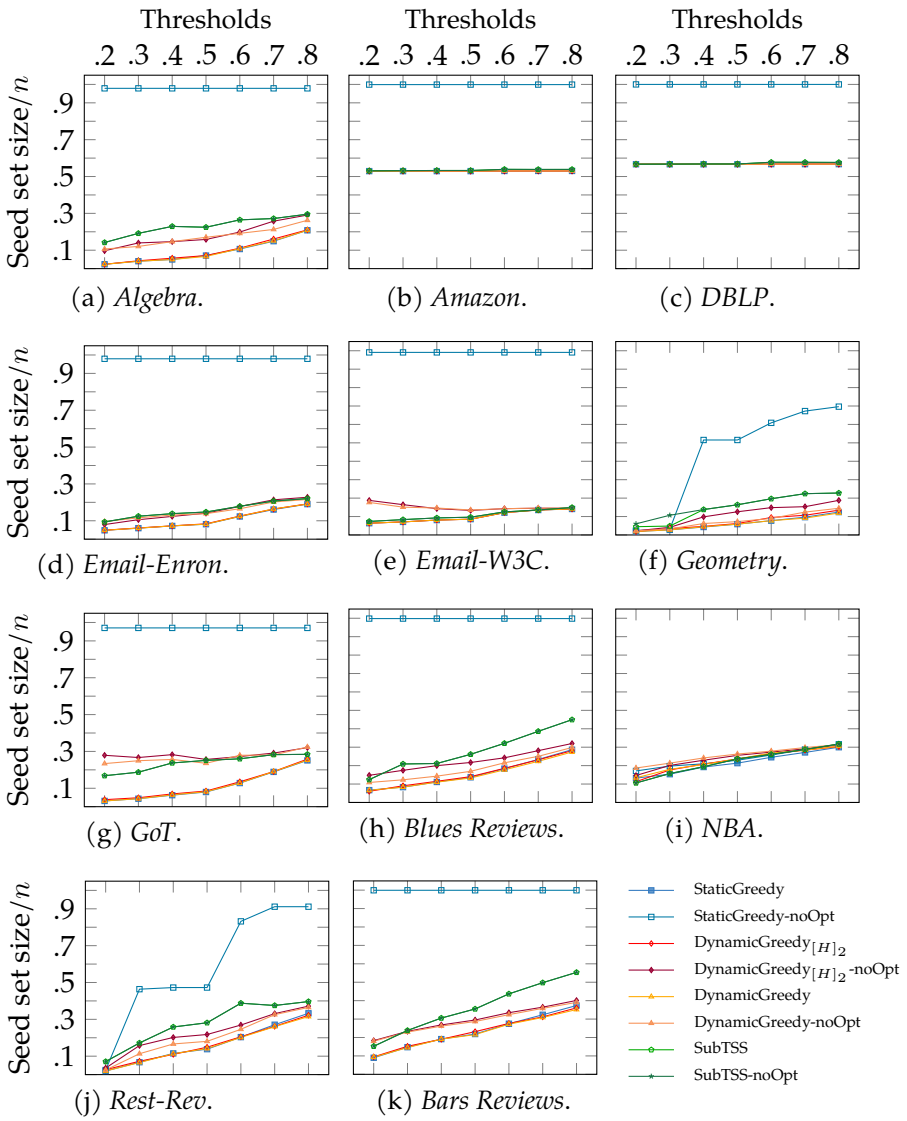


Figure 8.11: Scenario 4 - Ratio of the seed set size over the total number of nodes n per hypergraph, varying each node threshold proportionally - from 0.2 to 0.8 - to its degree, and using a majority policy for hyperedge thresholds.

step - retrieve the smallest seed sets and the three particular cases of the *Amazon*, *DBLP*, and *Email-W3C* hyper-networks where all heuristics (but StaticGreedy without optimization) achieve very similar results. Generally, as also observable in Figure 8.8, fewer vertices are required to influence the whole network when a majority policy on hyperedges is used. This outcome means that, on average, hyperedges with a higher threshold make the problem harder.

8.6.3 Execution Time Comparison

Having investigated the effectiveness of each heuristic in finding the smallest target set able to influence the whole network, we then analyzed their performance in terms of execution time. Specifically, we used the same setting of Scenario 1 using real-world hyper-networks, fixing random thresholds for both vertices and hyperedges. We simulated the execution of each heuristic 50 times, using the Julia package BenchmarkTools.jl. We run each experiment on an Ubuntu 18.04.2 LTS machine, equipped with Intel® Xeon® CPU E5-2660 0 @ 2.20 GHz and 32 GB RAM.

Figure 8.12 presents the distribution of the time (in seconds) required for each heuristic to complete the task. The first interesting outcome here, although not surprising, is that additive heuristics consistently need more time than the subtractive technique to accomplish the task. As already discussed in Scenario 1 (real-world networks), this behavior is due to the fact that additive techniques select many unnecessary vertices demanding a significant workload for the optimization phase, thus increasing the overall time needed to compute the final seed set. As expected, the additive dynamic heuristics - DynamicGreedy$_{[H]_2}$ and DynamicGreedy - require the highest computational time, as their algorithms, at each iteration, compute the residual network, updating the vertices candidate set. Generally, the SubTSSH subtractive heuristic requires at most half of the time of the dynamic additive heuristics to complete, thus representing a practical alternative when dealing with bigger hyper-networks.

Figure 8.13 shows the same results from a different perspective: in this

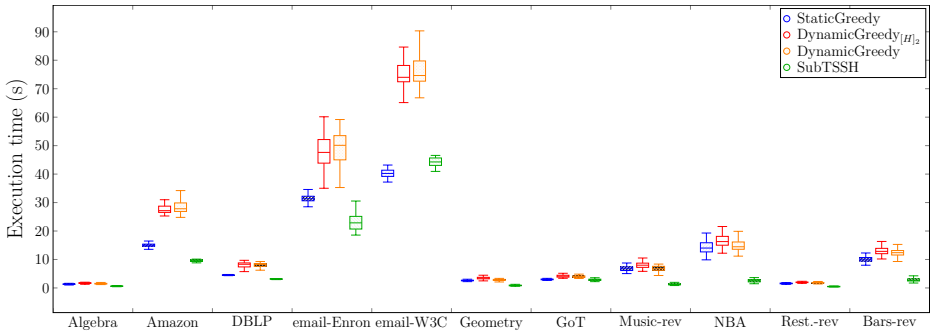


Figure 8.12: Distribution of the time (in seconds) required for each heuristic to complete.

case, the time needed for each heuristic to complete the task is plotted against the network size, evaluated as $|V| + |E|$. Thus, it allows us to analyze how each algorithm scales with increasing hyper-network size. This time, we also show the results associated with the four greedy strategies - StaticGreedy, DynamicGreedy$_{[H]_2}$, DynamicGreedy, and SubTSSH - run without the optimization phase. On the y-axis, the average running time of each heuristic is reported. Both axes have a logarithmic scale. Consistently with the data shown in Figure 8.12, SubTSSH followed by the optimization procedure achieves a very similar or even better performance of all the three additive heuristics run with the optimization phase. The StaticGreedy procedure run without optimization obtains the best time in absolute. However, as shown in Figure 8.14, it selects the majority of the vertices as seed set; thus, retrieving a useless solution in practice. That leaves SubTSSH executed without the optimization phase as the winner. As already discussed in the experiment of Scenario 1 (real-world networks), the optimization phase has a very low, if not zero, impact on reducing the initial seed set evaluated by the SubTSSH heuristic. Hence, the optimization phase may also be skipped when using this algorithm by paying a minimal price on the final seed set's dimension but still gaining execution time.

Figure 8.14 zooms into each data set analyzed, comparing the aver-

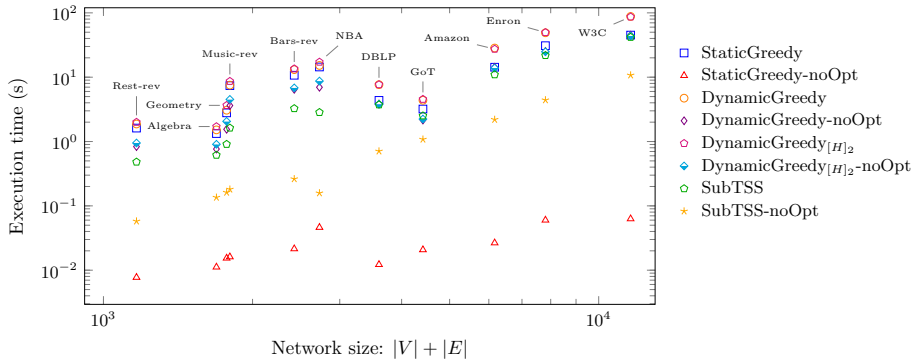


Figure 8.13: Distribution of the time (in seconds) required for each heuristic to complete the task against the size of the network ($|V| + |E|$).

age completion time of each heuristic against the size of the seed set evaluated. As expected, DynamicGreedy and DynamicGreedy$_{[H]_2}$ executed without the optimization phase need up to half of the time to complete than their corresponding optimized version. Obviously, the target set evaluated is larger. As previously discussed, the non-optimized version of StaticGreedy basically selects all vertices. This result suggests that TSSH heuristics designed for real-world networks should consider both the dynamicity of the influence process and the role that hyperedges play in the whole diffusion mechanism. Further, this experiment clarifies that SubTSSH reaches the best trade-off between the quality of the solution and the time needed to compute it.

8.7 Remarks

This chapter discussed a generalization of the linear threshold (LT) diffusion model on hypergraphs that notably differs from the corresponding model on graphs as it also involves hyperedges in the diffusion process. Hence, it enables the direct modeling of complex group dynamics happening in high-order networks. The Target Set Selection (TSS) problem is

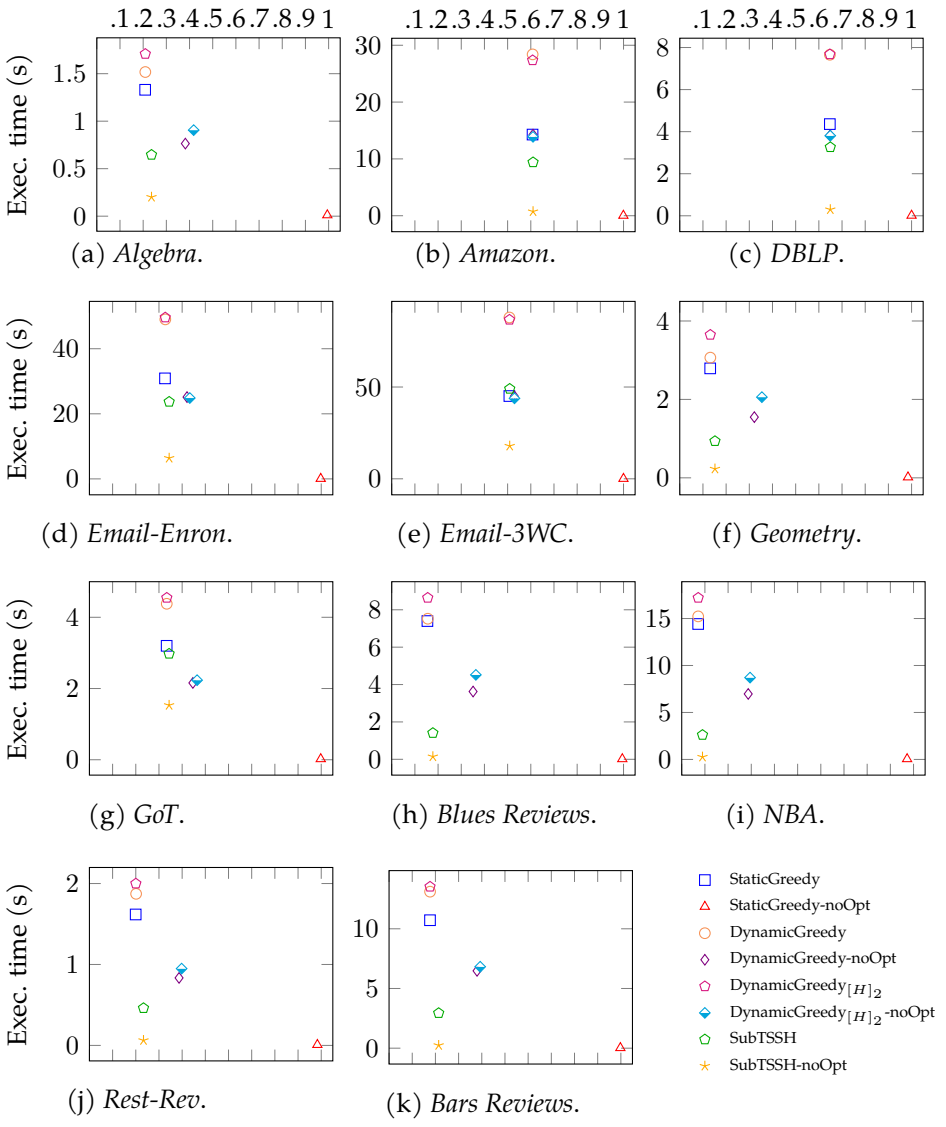


Figure 8.14: Average execution time vs seed set size.

a challenging puzzle arising in this domain. It consists in determining the

smallest subset of nodes (*seed-set*) able to influence the whole network. In our work, we generalized the TSS problem formalized for graphs to hypergraphs (TSSH problem), exploiting the proposed LT diffusion model for such structures.

To address the TSSH problem, we proposed three greedy-based additive heuristics and a subtractive procedure. All the described algorithms, being greedy-based approaches, do not guarantee that the cardinality of the seed set is (an approximation of) the optimal solution. For this reason, we also introduced an optimization step to improve the original solution found by each heuristic. We evaluated the effectiveness of the proposed heuristics considering the cardinality of the solution obtained and their execution time. We run extensive experiments on several random and real-world hyper-networks, varying the generative model and the vertex thresholds in the former case, the activation thresholds for both nodes, and hyperedges in the latter.

Results showed two different pictures according to the specific nature of the examined hyper-networks.

- In the case of random networks, we can summarize two significant points. First, no heuristic profoundly outperforms the others; second, the optimization procedure, whose objective is to remove unnecessary nodes from the seed set, does not impact the solution initially evaluated by each heuristic. An exception is made for random hypergraphs generated with the preferential attachment rule. In this case, the comments made for real-world hyper-networks hold.

- Regarding real-world networks, we can note that, although additive procedures generally provide worse results initially, they particularly benefit from the use of the optimization procedure. This process translates into better results than the subtractive procedure (for which the optimization procedure is practically irrelevant). Hence, this outcome depicts a completely reversed picture of random networks.

- Looking at execution times, we can observe how, although efficient, the optimization procedure harms the additive approaches' running time, thus making those heuristics more time-consuming.

Loosely speaking, comparing the seed set size obtained on different hypernetworks, it is possible to observe how generally the results are pretty heterogeneous and strictly depend on the networks' characteristics, such as the density of the network or the distribution of both node degrees and hyperedge size. In general, pure additive approaches may be used when the underlying network has either a random, $k-$ uniform or $d-$ regular structure. The pure subtractive approach of SubTSSH is suggested when dealing with real-world networks instead. Clearly, the network size is another critical aspect to consider when choosing which heuristic to apply.

PART

IV

Epidemics on High-order Temporal Networks

Summary

9

Epidemic Dynamics on Hypergraphs

> No great discovery was ever made without a bold guess.
>
> Isaac Newton

In short

This chapter and Chapter 10 presents the third and last key contribution of this dissertation: the formal definition of time-varying hypergraphs (TVHs), the introduction of direct and indirect interactions when studying an epidemic spreading via a TVH contact network, and an epidemic diffusion algorithm built on top of TVHs and direct and indirect contagion pathways. Specifically, we first motivate why using (temporal) hypergraphs rather than (temporal) graphs to analyze epidemic spreading processes (see Section 9.1) is beneficial. We then introduce the formal definition of temporal hypergraphs, describe a high-order SIS compartmental equation model suitable for TVHs, and discuss how we assem-

bled these elements into an agent-based framework (see Section 9.2). We further review how our work differentiates from the state-of-the-art (see Section 9.3), describe the experiment setting (see Section 9.4), and finally present a sensitivity analysis of the TVH model to the epidemic parameters and different discretization of the time intervals when direct or indirect contacts may happen (see Section 9.5).

The work described in this chapter has been presented in the following article:

- A. Antelmi, G. Cordasco, C. Spagnuolo, and V. Scarano. 2020. A Design Methodology for Epidemic Dynamics via Time-Varying Hypergraphs. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'20). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 61–69.

9.1 Motivation

Propagation of contagious diseases is a complex dynamic process that holds abounding human behavior aspects. Thus, modeling tools with a high level of expressiveness are required to resemble natural diffusion dynamics correctly.

Most of the well-known models adopted to analyze an epidemic diffusion and evaluate the outcomes of control policies are based on math equations (Equation-Based Models, EBMs) that have proved their ability to mimic the epidemic spreading in individuals [97, 89, 70]. However, these models assume that the population behavior and individual contact types are homogeneous [27]. The aforementioned is a severe limitation for real-world scenarios as it reduces the modeling effectiveness in describing different individuals and social behaviors [109]. Further, EBMs do not provide an easy way to model different contacts [79]. In particular, many epidemic contagions operate in *direct* contagion (person-to-person infection) and *indirect* contagion (person-to-environment infection, e.g., via furnishings or clothing). In other terms, EBMs provide us quantita-

tive information about the number of infected individuals in the worst-case scenario.

Agent-Based Models (ABMs) support researchers in overcoming these abstractions. ABMs are a modeling tool that easily incorporates features related to population and society and are widely adopted to simulate human behaviors under specific conditions [140, 172], particularly in epidemiological studies [124]. This tool allows researchers to naturally include human mobility data to model human interactions between the environment or other individuals. Typically, many epidemic ABMs also exploit networks to define possible agent interactions. Such data can be retrieved from online social networks, where users share their real-time location (Foursquare), geo-tag media posts (Facebook and Instagram), or review businesses (Yelp). The growing popularity of these online platforms and the ubiquitous online access provide gold data for studying users' habits, lifestyles, and mobility patterns to include in ABMs.

Graphs usually abstract and formalize contact among agents simulated with an ABM; hence, they also allow a formal analysis of the diffusion mechanisms happening within the simulation. Nonetheless, graphs do not always represent a suitable structure to analyze epidemic dynamics. As already thoroughly discussed in Chapter 5, Bodò et al. [41] first proposed modeling communities as hyperedges, based on the concept that an actual model of an epidemic outbreak has to consider two factors: community structure and infection pressure. They translated this approach into practice using different contagion probabilities according to the place. In addition, they bounded the likelihood that a susceptible individual becomes infected in a unit to be not proportional to the number of infected individuals within that unit. The authors show that using a non-linear function to model the infection pressure is crucial to not overestimate the epidemic propagation. In that way, they demonstrated that graphs are not a well-suited structure to capture the many-to-many relationships that come into play during epidemic propagation processes.

Further, these structures are not expressive enough to easily simulate

direct and indirect contact among individuals[1]. In addition, graphs do not consider when a connection happens, which is crucial in epidemic dynamics [150]. A contact network can be easily extended to include the time dimension by using *Time-Varying Graphs* (TVGs) [52, 68], a variant of the graph model, where a link between two nodes is valid only for a given time interval. Nonetheless, a TVG is not a definitive solution as the epidemic spreading process may still be over-estimated for particular diseases. Individuals in the same place at the same time come in contact with each other; as a result, there will be a link per each connection. In that case, the information tying together a group of individuals simultaneously in a particular geo-location is lost.

All the above considerations suggest that modeling epidemic spreading processes taking into account information conveyed by high-order interactions (e.g., indirect contagion pathways or probability to become infected not proportional to the number of infected within a unit) is crucial to correctly estimate and resemble diffusion dynamics happening in the real world.

9.2 High-order Temporal Epidemic Dynamics

In this section, we formally introduce temporal hypergraphs and present how these structures can model mobility patterns as well as direct and indirect contagion pathways. We further describe the underlying idea of our ABM design methodology and introduce the nomenclature used throughout this chapter and Chapter 10. We finally discuss a high-order diffusion process based on the SIS compartmental model.

9.2.1 Time-Varying Hypergraphs

To better mimic an epidemic spreading, we extended the definition of TVGs, presented by Casteigts et al. in [52], to hypergraphs. Employing a TVH to describe a contact network enables us to minimize the effect of

[1]In principle, additional memory can be used to store this information; however, this choice actually corresponds to using hypergraphs.

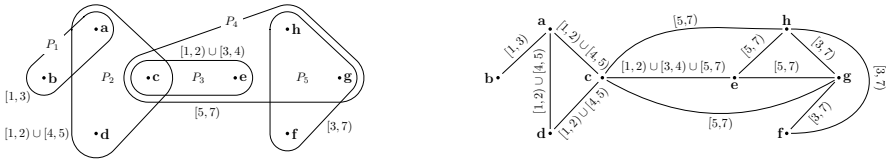
time and the presence of only direct contacts. Formally, a TVH is defined as follows.

Definition 9.1 : Time-Varying Hypergraphs (TVHs)

A *TVH* is a hypergraph $H = (\mathcal{V}, E, \mathcal{T}, \rho)$, where $\mathcal{T} \subseteq \mathbb{R}^+$ is the lifetime of the system and ρ: $E \times \mathcal{T} \rightarrow \{0, 1\}$ is an existence function, indicating whether a hyperedge exists at a given time.

For each $t \in \mathcal{T}$, we refer to the hypergraph $H_t = (\mathcal{V}, E_t)$ as the hypergraph corresponding to a particular time t, i.e., $E_t = \{e \in E : \rho(e, t) = 1\}$, where E_t denotes the set of hyperedges existing during t.

As defined by Bretto in [47] and described in Chapter 2, the two-section (or clique) representation of H, denoted with $[H]_2$, is a graph whose vertices are the vertices of H, and where two vertices form an edge if they are in the same hyperedge. Figure 9.1 presents an instance of a TVH (see Figure 9.1a) compared to its corresponding two-section graph (see Figure 9.1b). It illustrates a trivial TVH made up of 8 individuals (vertices), $\mathcal{V} = \{a, b, c, d, e, f, g, h\}$, and 5 geographical locations (hyperedges), $E = \{P_1, P_2, P_3, P_4, P_5\}$. Each hyperedge is labeled with its corresponding availability time interval(s) $[t_s, t_e) \subseteq \mathcal{T}$. It is worth noting that the $[H]_2$ representation introduces a loss of information in the con-



(a) TVH H.

(b) Graph $G = [H]_2$.

Figure 9.1: A simple TVH H (left) and its clique representation $G = [H]_2$ (right). Each (hyper)edge e is labeled with the corresponding availability time span, defined as $\cup\{t \in \mathcal{T} : \rho(e, t) = 1\}$.

tact network. For instance, it is not possible to recognize which is the time interval when the individuals c and e were both in the venue P_3 or P_4.

In our epidemiological modeling framework, we used the following definition of TVH with check-in function ω.

Definition 9.2 : TVH for an epidemic diffusion

A TVH for an epidemic diffusion is a hypergraph $H = (\mathcal{V}, E, \mathcal{T}, \omega)$, where

- $\mathcal{V}$ is the set of n vertices (users/agents);
- E is the family of m hyperedges (locations);
- $\mathcal{T} \in \mathbb{N}$ is the lifetime of the system[a];
- ω: $\mathcal{V} \times E \times (\mathcal{T} \times \mathcal{T}) \rightarrow \mathcal{T} \cup \{\perp\}$ is the function that maps an agent $v \in \mathcal{V}$ in a location $\ell \in E$ during the time interval $\tau = [t_s, t_e) \in (\mathcal{T} \times \mathcal{T})$, with $t_s \leq t_e$, to the last check-in time $t_{v,\ell}$ of v in ℓ in the specified interval. If v never checked-in in ℓ during τ, the function returns $\perp$.

[a] In practice, the lifetime $\mathcal{T}$ is a discrete valued interval delimited by two Unix timestamps.

The function ω only keeps track of the last check-in time $t_{v,\ell}$ for a vertex v in a given location ℓ during a time interval τ. If v did not checked-in in the same location ℓ or in another location ℓ' in the next time interval τ', the value of $t_{v,\ell}$ is still considered a valid check-in time in the current timestamp τ'. We store the last check-in time $t_{v,\ell}$ of a vertex v in a location ℓ as the weight of v in the hyperedge representing ℓ. This modeling strategy allows us to simulate direct and indirect contagion processes easily: we can effortlessly know where the user is during each time interval.

9.2.2 A Design Methodology

We based the idea of our ABM design methodology on the assumption that two spreading policies regularize an epidemic process: *direct* and *indirect* contagions between individuals and environments. Direct contact implies a pairwise interaction between two individuals in the same place. In contrast, an indirect contagion embodies the interchanges that may happen between agents and locations. These two types of contacts are a natural consequence of each person's daily activities and commuting routes. For instance, when an agent moves from its home to its workplace, it may be infected by touching some furniture or simply breathing contaminated air (indirect interaction) or by a face-to-face talk with another agent (direct exchange).

In the following section (see Section 9.2.3), we describe a diffusion algorithm whose spreading process is designed in a discrete-event fashion and exploits the TVH structure to discover whether direct or indirect interactions may happen. During each time interval, agents are simulated according to their scheduling policy. Then, our diffusion algorithm is performed. As agents are free to move, the epidemic has the chance of spreading from one location to another. Simultaneously, the outbreak may still spread across agents located in the same place at a particular time interval. While direct contamination requires agents' co-presence, indirect connections happen between agents and the environment, and co-presence is not needed.

The scheduling routine of each agent is defined by an input check-ins data set, describing where each agent is and when the agent entered that specific location. The simulation time is split into fixed-width intervals of length Δ, corresponding to when indirect contagion may happen. Given a sampling time Φ of a check-in data set, the total number of time intervals considered is $\lceil \frac{\Phi}{\Delta} \rceil$. In practice, all check-ins happening within the same location in a Δ time window are grouped together, and a hyperedge is added to the TVH to represent the location where a group interaction (or indirect contact) is happening. For each time window, if the

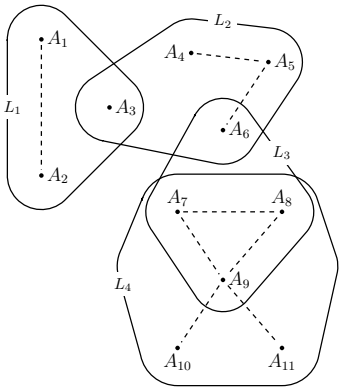
time difference between two check-ins in the same location is lower than a given small value δ, we consider direct contact happening between those two agents. This methodology approximates the notion of direct contact when high-granular data on the individual routine is not available. It is worth noting that to not erroneously discard direct contact happening in the time window δ between two consecutive time intervals of length Δ, each interval actually lasts $\Delta + 2\delta$. Hence, we generate overlapping intervals, with the interval τ overlapping the last δ window of the previous interval τ' and the first δ window of the following interval τ''.

Agents' scheduling policy

The current model does not directly take into account the order in which agents visit a location or their check-out time. Including such information in the diffusion process might lead to a more accurate representation of the epidemic dynamics. For instance, we could consider the exact time window within which two agents had direct contact as well as the permanence of an agent in a location. This consideration translates into modifying the direct and indirect infection probabilities to accommodate that a long permanence or a long chat may lead to higher infection probabilities. Unfortunately, there are no data sets reporting check-out information as (location-based) social networks usually only offer check-in functionalities. High-resolution mobility data sets deriving from other data sources also exist, but they are either too small [166] or proprietary [60]. Recently, various Data For Good Programmes (e.g., Meta [135]) began offering aggregated mobility data. However, this data cannot be directly exploited within our model as we need to know where and when a specific person has been. A possible solution is to design an ABM in which agents' routines mimic those mobility patterns. Usually, the parameters regulating such

models may grow even to hundreds [93, 7, 8] and, thus, require proper validation and calibration. Designing and building complex ABMs is an independent research field.

Figure 9.2 reports an instance of a TVH corresponding to the check-in data set listed in the table with 11 agents and 4 locations. In this example, the data is split into fixed-width intervals of length $\Delta = 3$ hours; hence, all hyperedges corresponding to a specific location indicate that indirect



Agent	Location	Check-in
A_1	L_1	8 : 00 : 00
A_2	L_1	8 : 00 : 49
A_3	L_1	8 : 30 : 00
A_3	L_2	9 : 00 : 00
A_4	L_2	9 : 30 : 00
A_5	L_2	9 : 30 : 40
A_6	L_2	9 : 31 : 10
A_6	L_3	10 : 00 : 00
A_7	L_3	9 : 00 : 00
A_8	L_3	9 : 00 : 10
A_9	L_3	9 : 00 : 15
A_7	L_4	9 : 30 : 00
A_8	L_4	9 : 40 : 00
A_9	L_4	10 : 00 : 00
A_{10}	L_4	10 : 00 : 10
A_{11}	L_4	10 : 00 : 20

Figure 9.2: Direct and indirect contacts modeled via a TVH.

contact is happening among the agents that have visited that location in a 3 hours time frame. Dashed lines indicate that direct contact is happening between two agents as they have been in the same location in a time frame of less than $\delta = 1$ minute.

Table 9.1 provides the definitions of the concepts adopted in the next sections of this chapter and Chapter 10. In the following, we will omit the subscript t when the simulation time is clear from the context.

Table 9.1: TVH related concepts and notation used in this chapter and Chapter 10.

Symbol/Concept	Description
$H = (\mathcal{V}, E, \mathcal{T}, \omega)$	TVH representing the data (see Definition 9.2.1).
Φ	Time span of the data.
Δ	Real value (minutes, hours or days) corresponding to the time discretization parameter. It further refers to the time span when *indirect* contagions may happen.
δ	(Small) Real - value (milliseconds, seconds or minutes) defining when direct infections may take place. A direct contagion is established if two agents stay in the same location within a time difference less than δ.
t	Current simulation time ($t \in \mathcal{T}$).
$\Gamma_t(a) = \{\ell \in E : \omega(a, \ell, (t_s, t_e)) \neq \perp\}$	Location function of an agent $a \in V$ in a given simulation time t. It returns the set of locations visited by a during the interval t.
$\mathrm{N}_t(a) = \bigcup_{\ell \in \Gamma_t(a)} V_t(\ell)$	Neighborhood function of an agent $a \in V$ in a given simulation time t. It returns the set of neighbors of a during the simulation time t, where $V_t(\ell)$ denotes the set of agents that visited the location ℓ during t.
$\Upsilon(a, \ell)$	Time function providing the last check-in time of the agent a in the location ℓ.
$\mathrm{T}_t(a)$	Infection state (1 infected, 0 not infected) of an agent in a given simulation time t.
$\mathrm{T}_t(\ell)$	Infection state (1 infected, 0 not infected) of a location in a given simulation time t.
$\mathrm{X}_t(a, b) = [\![\exists\, \ell \in \Gamma_t(a) \cap \Gamma_t(b) \wedge$ $\lvert\Upsilon(a, \ell) - \Upsilon(b, \ell)\rvert < \delta\,]\!]$	Direct contact function. Given two agents a and b, it returns 1 if they have direct contact in the time t; 0 otherwise.

9.2.3 Diffusion Process

We based the SIS model we present in this chapter on the work of Bodò et al. [41], in which the infection and recovery states are ruled by a Poisson process. Thus, either a susceptible individual or location becomes (directly or indirectly) infected with a probability $1 - e^{-\beta_x f(n)}$. Here, β_x denotes the infection rate per-contact (considering either direct or indirect contacts), n is the number of infected entities (individuals or locations), and $f(\cdot)$ is a function of n. Similarly, an infected agent or location recovers with probability $1 - e^{-\gamma_x}$, where γ_x denotes the recovery rate for agents and locations.

During each time step, our diffusion algorithm proceeds in three contagious phases.

1. *Agent-to-Environment*. The first phase simulates the *environment infectiveness*. For all non contaminated locations, (i.e., $\ell \in E : T_t(\ell) = 0$), we compute the number of infected agents that have visited that location in the current simulation step t:

$$I^e(\ell) = \sum_{a \in V(\ell)} \mathrm{T}(a).$$

This value is then used to update the infection state of a susceptible location ℓ, as expressed by the following rule:

$$\mathrm{T}_{t+1}(\ell) = \begin{cases} 1, & \text{infected with probability } 1 - e^{-\beta_e f^e(I^e(\ell))} \\ 0, & \text{not infected otherwise} \end{cases},$$

where β_e is the infection rate of the locations, and $f^e(\cdot)$ is a non-linear function, typically adopted to govern the behavior of the epidemic outbreak over hyperedges [41]. In our experiments (see Section 9.5 and Section 10.4), we considered the following regularization function:

$$f^e(x; c) = \begin{cases} x, & \text{if } 0 \leq x \leq c \\ c, & \text{if } x > c \end{cases},$$

where c is a constant given as parameter.

2. *Agent-to-Agent*. The second phase simulates the *direct* propagation process. For all susceptible agents (i.e., $a \in V : T_t(a) = 0$), we compute the total number of infected neighbors in the current simulation interval t. Formally,

$$I^d(a) = \sum_{b \in \mathrm{N}(a)} \mathrm{T}(b)\mathrm{X}(a,b).$$

This value is then used to update the infection state of a susceptible agent a, as expressed by the following rule:

$$\mathrm{T}_{t+1}(a) = \begin{cases} 1, & \text{infected with probability } 1 - e^{-\beta_d I^d(a)} \\ 0, & \text{not infected otherwise} \end{cases},$$

where β_d is the infection rate of the agents due to direct contagions.

3. *Environment-to-Agent.* The third and last phase simulates the *indirect* propagation process. For all susceptible agents, (i.e., $a \in V : T_t(a) = 0$), we compute the number of infected locations visited in the current simulation interval t. Formally,

$$I^i(a) = \sum_{\ell \in \Gamma(a)} \mathrm{T}(\ell).$$

This value is then used to update the infection state of a susceptible agent a, as expressed by the following rule:

$$\mathrm{T}_{t+1}(a) = \begin{cases} 1, & \text{infected with probability } 1 - e^{-\beta_i I^i(a)} \\ 0, & \text{not infected otherwise} \end{cases},$$

where β_i is the infection rate of the agents due to indirect contagions.

We consider the simulation proceeding in discrete steps. At each simulation interval t, every agent independently runs its *step function* and up-

dates its internal state, which will be effective in the next simulation phase $t + 1$.

Algorithm 6 reports the pseudo-code of the ABM simulation, specifically the high-order SIS compartmental diffusion model devised for hypergraphs. First, the algorithm computes the simulation intervals as described in Section 9.2.2. Then, at each simulation step, it computes the hypergraph corresponding to the current timestamp (line 2), and it runs the three epidemic diffusion phases. In the first phase, *Agent-to-Environment* (lines $3-8$), the infection status of every location is updated. In the second phase, *Agent-to-Agent* (lines $9 - 12$), the epidemic is propagated through the network using direct contacts, and the infection status of the agents is updated. Finally, in the last phase, *Environment-to-Agent* (lines $13 - 18$), the epidemic is propagated using indirect contacts, and the infection state of the agents is updated again. The recovery process is computed only once for both agents and locations.

This diffusion algorithm can be easily extended by implementing additional phases (before or after the algorithm execution) describing supplementary agent behaviors. Therefore, the algorithm is suitable for other typologies of ABMs. For instance, to simulate the introduction of the non-pharmaceutical interventions described in Chapter 10, we extended the diffusion algorithm to accommodate the specific details of each intervention and model additional behaviors. All the parameters in input to our model are specified in Section 9.4.2.

Algorithm 6 High-order SIS compartmental model on TVHs

for $t \in$ compute_intervals() **do**
  $\mathcal{H}(\mathcal{V}, E) \leftarrow$ compute_tvh(t)
  for $\ell \in E$ **do** ▷ Agent-to-Environment.
    if $\mathrm{T}_t(\ell) = 0$ **then**
      if rand() $< 1 - e^{-\beta_e f^e(I^e(\ell))}$ **then** ▷ rand() generates a random number in $[0, 1]$.
        $\mathrm{T}_{t+1}(\ell) \leftarrow 1$
    else if rand() $< 1 - e^{-\gamma_e}$ **then**
      $\mathrm{T}_{t+1}(\ell) \leftarrow 0$
  for $a \in \mathcal{V}$ **do** ▷ Agent-to-Agent.
    if $\mathrm{T}_t(a) = 0$ **then**
      if rand() $< 1 - e^{-\beta_d I^d(\ell)}$ **then**
        $\mathrm{T}_{t+1}(a) \leftarrow 1$
  for $a \in \mathcal{V}$ **do** ▷ Environment-to-Agent.
    if $\mathrm{T}_t(a) = 0$ **then**
      if rand() $< 1 - e^{-\beta_i I^i(\ell)}$ **then**
        $\mathrm{T}_{t+1}(a) \leftarrow 1$
    else if rand() $< 1 - e^{-\gamma_a}$ **then**
      $\mathrm{T}_{t+1}(a) \leftarrow 0$

9.3 State-of-the-art Comparison

In our work, we adopt temporal hypergraphs to correctly model an epidemic propagation in a many-to-many fashion and capture that people, moving through different locations, form a community in a given time and space. Alike (temporal) graphs, TVHs abstract and formalize contact among agents simulated with an ABM, but adding information about where the contact is happening. Hence, such structures allow to formally analyze diffusion mechanisms while accounting for group interactions and indirect contagion processes via contaminated locations.

The SIS epidemic propagation model we propose fits within the framework defined by Bodò et al. [41]. As the authors, we also modeled infec-

tion and recovery as Poisson processes; further, we used the same non-linear function to model the infection pressure. This choice allows us to resemble the situation in which if an infected individual contaminates an object or surface, then further contamination by other individuals is less relevant. In this case, doubling the number of infected users will increase the risk by a factor of less than two.

Starting from these common elements, our model extends their work by explicitly accounting for both direct and indirect contacts in the spreading process (see Section 9.2.2). In our diffusion model, an individual may become infected not only because of close interaction with another sick person but also because the same individual may have visited a contaminated location (e.g., touching infected furniture in a restaurant). Vice versa, a location (represented by a hyperedge) may become contaminated (and thus spread the epidemic) as infected individuals may have visited that place. This aspect represents the first key feature of our spreading model: both vertices (people) and hyperedges (locations) actively engage in the spreading process, and the epidemic bounces from one to the other (see Section 9.2.3). The second main characteristic of our model is the temporal component. We do not assume a static contact network; instead, we formally capture people's routine via temporal hypergraphs to resemble real-world commuting patterns (see Section 9.2.1).

The features mentioned above characterize and distinguish our work from all the models discussed in Chapter 5. The first difference lies in the nature of the hypergraphs analyzed, as all related literature deals with static hyper-networks. The second point concerns when a hyperedge is considered infected and, potentially, infectious. In our model, a location may become contaminated and become infectious according to Poisson processes proportional to the number of infected nodes within that hyperedge. This concept also differs from the models of Jhun et al. [106] and de Arruda et al. [61], where a hyperedge of degree d becomes infectious if $d-1$ nodes within it are infected [106] or if they are higher than a given threshold [61]. Contrarily, Landry et al. [122] do not have the concept of hyperedge infectiousness. In their work, a susceptible node in

a hyperedge may become infected according to a specific spreading policy defining the minimum number of nodes in the same hyperedge that should be infected. Similarly, Suo et al. [173], and Ma et al. [134] do not explicitly model the concept of group infectiousness or even peer pressure. These works investigate compartmental models on hypergraphs in the context of rumor spreading and information diffusion, but they only use the hypergraph structure to model the situation in which a person may spread information to a whole group. The work of Higham et al. [92] was only published in August 2021, after our research. They inherit the model of Bodò et al. and specifically focus on studying spectral conditions characterizing whether the disease vanishes.

9.4 Experiment Setting

This section describes the experimental setup of the simulation, introducing the underlying epidemiological assumptions, detailing the simulation parameters, and analyzing the data set used.

Code Availability

The model and the experiments are implemented in Julia, exploiting the library `SimpleHypergraphs.jl` (see Chapter 7). The code is open-source and available at the following GitHub public repository: `https://github.com/alessant/HGEpidemics`.

9.4.1 Assumptions

As mentioned in Section 9.2, the proposed epidemic design methodology is based on high-order relationships that may happen during an epidemic outbreak between humans and environments via direct and indirect contagious pathways. From the epidemiological point of view, it is crucial to

clarify the assumptions lying underneath our model. First, the diffusion procedure assumes that all infected individuals are asymptomatic. This choice advantages the epidemic spreading as all infected propagate the infection, representing an epidemic outbreak's optimal case. Second, we do not consider the notion of incubation within our model, intended as the time elapsed between exposure to a pathogenic organism and when symptoms and signs are first apparent. In our model, each agent contracts the infection according to a transmission probability and proportionally to the number of infected individuals it met or the number of locations it visited. As for the previous point, this choice helps the epidemic spread out as the person becomes sick immediately. Third, we do not study how the time-length individuals spend together plays a role in the infection dynamics (see Section 9.2.2). Last, we fixed the value of each epidemic parameter during the whole simulation steps.

9.4.2 Simulation Parameters

Table 9.2 lists all the simulation parameters used to control the epidemic spreading. In the experimental part of this chapter, we study the model's sensitivity to the epidemic parameters and different discretization of the time intervals when direct or indirect contacts may happen. We modify the parameter configuration according to the specific scenario to simulate.

We generated the population mobility pattern according to the most crowded month (May 2012) of the Foursquare data set (see Chapter 6). As the fraction of infected results from probabilistic processes, we ran each simulation scenario 80 times, considering the averaged value as a result. The observed variance was negligible. The simulation is initialized with 20% of infected agents.

Modeling real-world epidemics

As discussed in Section 9.2.2, to test this approach to model real-world epidemics (e.g., Covid-19), we need detailed mobility data or a validated ABM resembling people's mobility patterns. Then, the epidemic diffusion model can be calibrated on top of this data and its compartments modified to accommodate the nature of the pathogen. Testing the so-obtained composite model against real-world data usually means comparing the simulated versus the real epidemic curve.

Table 9.2: Simulation parameters. For each parameter, the table reports: a short description, its domain, and its value in the simulation (*variable* indicates that the parameter changes according to the specific experiment).

Parameter	Description	Domain	Value
β_d	Probability that an agent is infected by another agent via direct contact in *Agent-to-Agent*.	$[0, 1]$	Variable
β_i	Probability that an agent is infected via indirect contact due to a location in *Environment-to-Agent*.	$[0, 1]$	Variable
β_e	Probability that a location is infected by an agent in *Agent-to-Environment*.	$[0, 1]$	Variable
γ_a	Probability that an agent spontaneously recovers.	$[0, 1]$	0.1
γ_e	Probability that a location is sanitized.	$[0, 1]$	0.06
c	Number of contacts in *Agent-to-Environment*.	$\mathbb{N}$	5
Δ	Time window within which indirect contagion may happen.	$\mathbb{R}$	Variable
δ	Time window within which direct contagion may happen.	$\mathbb{R}$	Variable

9.4.3 The Foursquare Data Set

Before delving into analyzing the TVH model's sensitivity and using a reasonable parameter value for δ, we investigated the check-ins distribution in the most crowded week in the ten months available of the Foursquare data set ($7 - 14$, May, 2012).

We estimated the value of the parameter δ, i.e., the time window within which a direct contagion may occur, by examining the time elapsed between two check-ins that happened in the same place. Figure 9.3 presents this distribution. Interestingly, but not surprisingly, the data contain some time windows where no check-ins are available, translating into the fact that the epidemic has no probability of propagating during those specific intervals. Generally, most of the intervals exhibit the same trend with a median value of about 1 hour (meaning that the average distance between two check-ins in the same place is 1 hour).

Having estimated $\delta = 1$ hour, we inspected the number of direct contacts within each place using this value. Figure 9.4 shows this distribution per time interval. Also in this case, we can note a homogeneous trend with direct contacts being evenly distributed over the whole week, with a median value of about $1,750$.

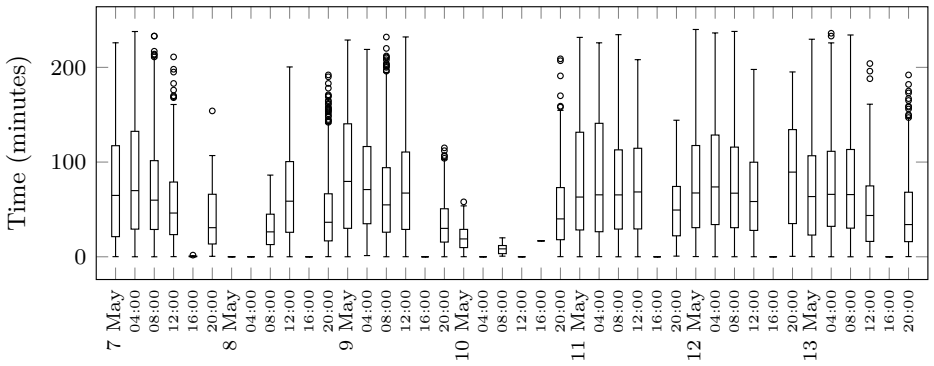


Figure 9.3: Distribution of the time elapsed between two check-ins in the same place over a period of 7 days, fixing $\Delta = 4$ hours.

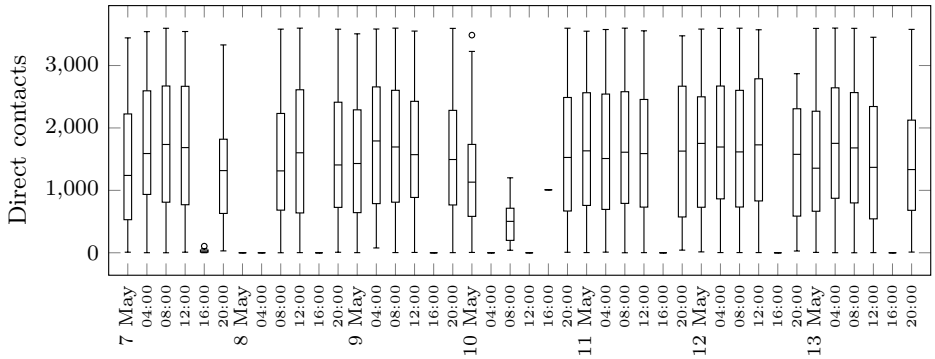


Figure 9.4: Number of *direct* contacts for each person in 7 days, fixing $\Delta = 4$ hours and $\delta = 1$ hour.

Finally, we investigated the distribution of indirect contacts, namely, the number of different locations each user has visited. Figure 9.5 reveals how many different places users have visited within each time interval. The plot shows that users tend to visit (or, at least, checking-in) just one venue or a limited number of them. Several outliers visit nearly 20 venues but no more than 30. As described in various works [189, 84, 19], this may represent a typical kind of power-law behavior, where only a few users post the majority of the online content.

9.5 Sensitivity Analysis

To evaluate the proposed approach and to what extent the described design methodology can resemble the epidemic spreading, we developed an ABM simulation running the SIS model over a population of individuals. Specifically, we tested the model expressiveness in distinguishing the epidemic diffusion via direct and indirect contagion pathways (see Section 9.5.1). We further analyzed the effect of time when modeling contacts by varying the values of Δ and δ. This experiment allowed us to explore the dynamic of the epidemic by increasing or decreasing the amount of direct and indirect contacts (see Section 9.5.2).

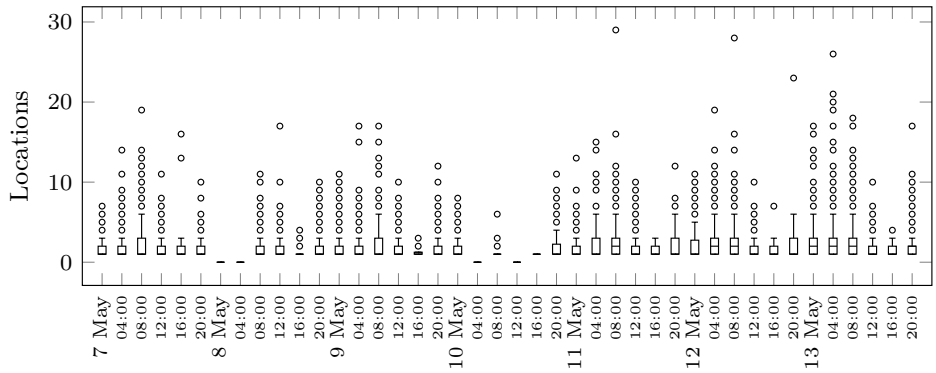


Figure 9.5: Number of *indirect* contacts (or location visited) for each person in 7 days, fixing $\Delta = 4$ hours.

9.5.1 Direct vs Indirect Contagion

In this first experiment, we tested the sensitivity of the TVH-based model to the epidemic parameters; in other words, we analyzed the model's ability in mimicking the epidemic spreading according to direct and indirect contagion pathways. We run the agent-based simulation according to two different parameter configurations (see Section 9.4.2). In the first configuration (*Low*), we used $\beta_d = 0.2$, $\beta_i = 0.1$, $\beta_e = 0.06$, $\gamma_e = 0.10$, $\gamma_a = 0.06$, and $c = 5$. In the second configuration, (*High*), we used $\beta_d = 0.8$, $\beta_i = 0.4$, $\beta_e = 0.26$, $\gamma_e = 0.10$, $\gamma_a = 0.06$, and $c = 5$. As rule of thumb, we selected the value β_d and computed $\beta_i = \frac{\beta_d}{2}$, and $\beta_e = \frac{\beta_d}{4}$. We set $c = 5$, based on the work of Bodò et al. [41]. We fixed the value of $\Delta = 4$ hours and $\delta = 1$ minute, selecting these values according to a realistic spreading policy for airborne disease transmission. We run three scenarios for each configuration to consider the pathogen spreading (*i*) only via direct contact, (*ii*) only through indirect contact, or (*iii*) using both contagion routes. Table 9.3 summarizes the parameter configuration of each scenario. In all settings, the average time an agent is infected is $16.6 \cdot \Delta$ intervals (2.7 days on average), while for a location is $10 \cdot \Delta$ intervals (1.7 days on average).

Figure 9.6 illustrates how the fraction of infected agents evolves ac-

Table 9.3: Parameter configuration for all simulation scenarios.

Scenario	β_d	β_i	β_e	γ_a	γ_e	Δ	δ
Low: Direct & Indirect	0.20	0.10	0.06	0.06	0.10	4 hours	1 minute
Low: Direct	0.20	0.00	0.00	0.00	0.10	4 hours	1 minute
Low: Indirect	0.00	0.10	0.06	0.06	0.10	4 hours	1 minute
High: Direct & Indirect	0.80	0.40	0.26	0.06	0.10	4 hours	1 minute
High: Direct	0.80	0.00	0.00	0.00	0.10	4 hours	1 minute
High: Indirect	0.00	0.40	0.26	0.06	0.10	4 hours	1 minute

cording to the SIS spreading policies just defined. The plot reveals an interesting pattern in the epidemic diffusion. In both parameter configurations (*Low* and *High*), the trend of infected agents obtained when using both direct and indirect contagion routes is not statistically different from the series obtained considering only indirect contagion pathways. The two lines on the top and the two in the middle represent this behavior. On the contrary, when looking at the epidemic trend conveyed by only direct contacts (the two lines at the bottom), we can note how the diffusion will eventually drop out in both configuration scenarios.

The nature of the data well explains this outcome. As discussed in Section 9.4.3, the Foursquare data set is highly sparse by nature, and the majority of the check-ins refer to common places like transportation or general entertainment (for instance, the 41 most crowded locations in the data set refer to the transportation system). Consequently, these locations have the potential to spread the epidemic across a considerable number of agents. On the other hand, the data set does not convey enough information about direct contacts happening within each location as we can only access the information that a Foursquare user checked in a specific location in a given time. This finding reveals a critical insight that should be accounted for when studying epidemic dynamics: indirect contact plays a critical role in spreading the epidemic. Hence, indirect contagion routes cannot be overlooked in scenarios in which epidemic diffusion processes do not necessarily involve direct contact between two individuals as one

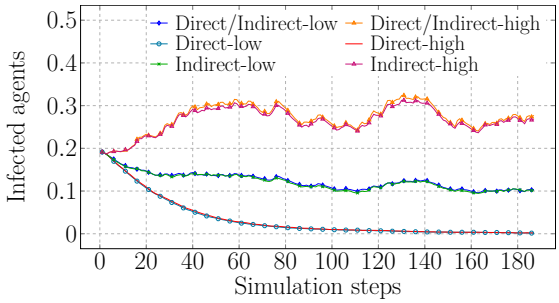


Figure 9.6: Epidemic spreading over a period of 30 days, considering contagion due to (*i*) both direct and indirect contacts, (*ii*) direct contacts, and (*iii*) indirect contacts.

location can potentially infect many people.

9.5.2 Modeling the Effect of Time

The epidemic diffusion process and its impact on the population are strictly related to the pathogens' life cycle and survival time in the environment. In this experiment, we investigated how discretizing the time window within which a direct (δ) or indirect (Δ) contagion may happen impacts the epidemic spreading in the network. We devised several simulation scenarios in which we ranged the value of δ from 1 to 60 minutes and the value of Δ from 4 to 24 hours. It is worth noting that smaller values of δ and Δ correspond to higher accuracy in computing direct and indirect contacts, respectively. We used the parameter configuration *Low* for all scenarios (see Table 9.3).

Figure 9.7 details how the fraction of infected agents changes according to the different time intervals used to model contacts. The plot in the top-left corner of the image shows the most accurate configuration ($\delta = 1$ minute and $\Delta = 4$ hours), while the plot in the bottom-right corner the less accurate. Looking at the figure from left to right, we can observe that direct contagion pathways become more relevant for the epidemic diffusion as the value of δ increments. However, initializing δ with a value

too large would mean evaluating too many direct contacts compared to a real-life system. Therefore, this would mean over-estimating the epidemic propagation and, in practice, modeling the contact network as a graph rather than a hypergraph. Similarly, looking at the figure from top to bottom, we can observe that the higher the value of Δ, the higher the contribution of indirect contacts to the epidemic diffusion. As happens for the parameter δ, increasing the value of Δ implies computing too many indirect contacts; hence, assuming a longer pathogens life. The outcome of this experiment suggests that determining the proper values of δ and Δ according to given disease properties is fundamental to correctly estimate the epidemic propagation.

9.6 Remarks

Propagation of contagious diseases is a complex dynamic process that holds abounding human behavior aspects: to correctly resemble real diffusion dynamics, modeling tools with a high expressiveness are required. Towards this direction, this chapter discussed a novel approach to model epidemic propagation, taking into account the high-order interactions happening in the real world. Specifically, we proposed:

- The introduction of direct (human-to-human) and indirect (human-to-environment) interactions when studying an epidemic spreading;
- The formal definition of temporal hypergraphs to abstract such high-order relationships;
- An epidemic diffusion algorithm on TVHs considering both direct and indirect contagion pathways;
- The development of the SIS compartmental equation model into an ABM that exploits our methodology to simulate interactions between agents and locations.

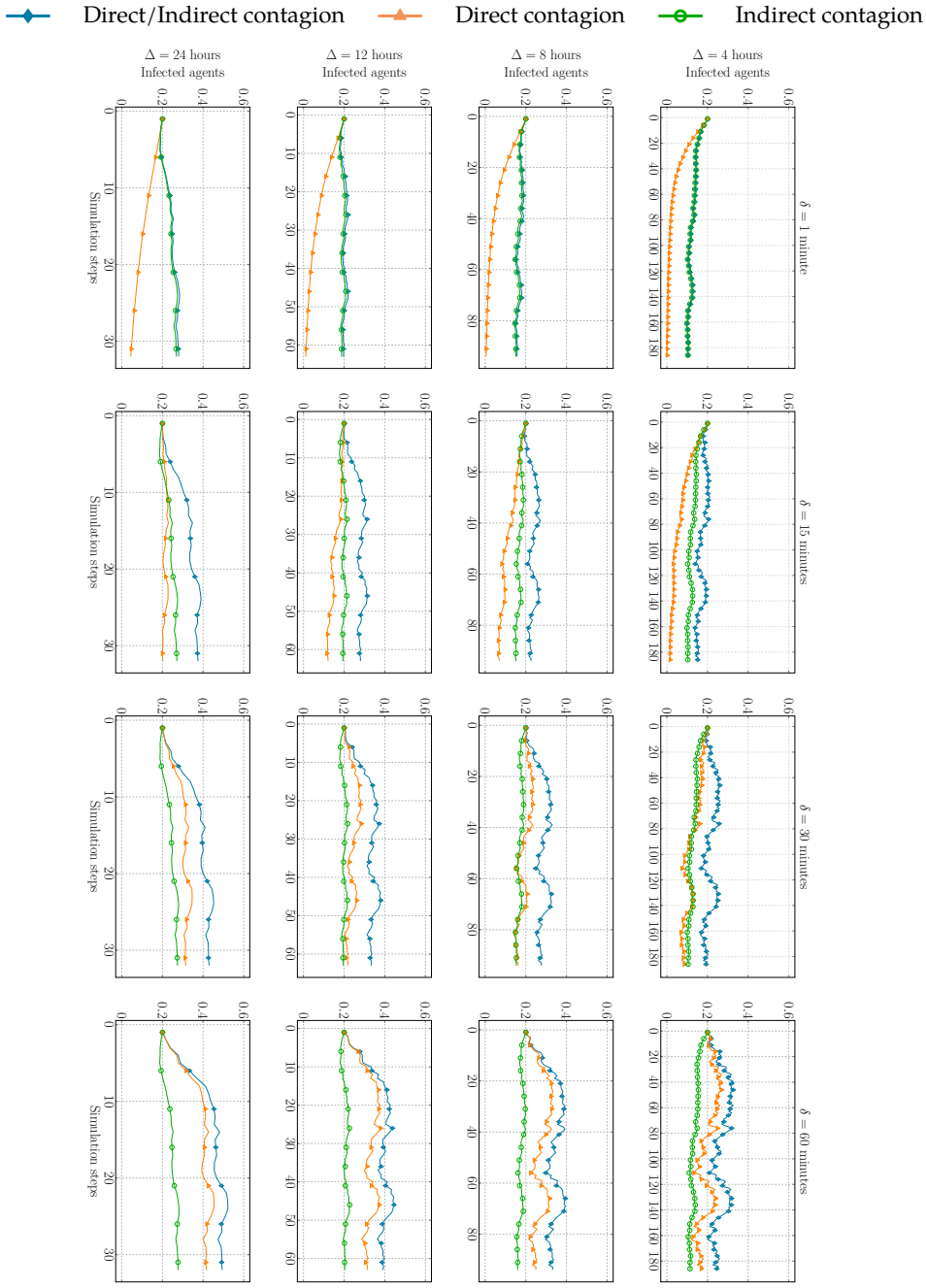


Figure 9.7: Epidemic evolution over 30 days, varying the interval lengths of Δ (vertically) and of δ (horizontally).

To evaluate the proposed approach and to what extent the described design methodology could resemble the epidemic spreading, we (i) tested the model expressiveness in distinguishing the epidemic diffusion via direct and indirect contagion pathways, and (ii) we analyzed the effect of time when modeling contacts by varying the values of Δ and δ. The results obtained demonstrated that the TVH-based model is sensitive to variations in the input parameters. Hence, it can be safely calibrated with accurate epidemiological data to study spreading dynamics. The results further suggest that:

- Exploiting TVHs may improve the accuracy of the estimation of an epidemic diffusion. For instance, as discussed in Section 9.5.1, indirect contacts may be a strong vehicle of contagion and actually govern the epidemic trend. As a consequence, indirect contagion routes cannot be overlooked in scenarios in which epidemic diffusion processes do not necessarily involve direct contact between two individuals as one location can potentially infect many people;
- Correctly modeling the time interval within which direct or indirect contagion may happen is critical to properly resemble real diffusion dynamics and not overestimate either contagion type as seen in Section 9.5.2.

10

Modeling Epidemic Control Strategies

> The problem is not the problem. The problem is your attitude about the problem.
>
> *Jack Sparrow*
> The Pirates of the Carrabean

In short

Built on top of the TVH model presented in Chapter 9, this chapter discusses a fine-grain modeling methodology for Non-Pharmaceutical Interventions (NPIs), based on high-order relationships between people and environments, mimicking direct and indirect contagion pathways over time. In this chapter, we first motivate why evaluating such epidemic control strategies in the framework of ABMs and high-order interactions (see Section 10.1). We then delve into reviewing personal protective, environmental, and social distancing measures and how they can be embedded

into an epidemiological model based on high-order networks, ABMs, and the SIS equation-based model (see Section 10.2). We further describe how we formally enriched the TVH modeling framework to support the evaluation of NPIs, and examine how our work differentiates from the state-of-the-art (see Section 10.3). After discussing the experiment setting (see Section 10.4) and assessing the ability of each intervention in controlling an epidemic propagation (see Section 10.5), we discuss a multi-objective optimization framework, which, based on the epidemiological data, calculates the NPI combination that should be implemented to minimize the spread of an epidemic as well as the damage due to the intervention (see Section 10.6). We finally examine the most interesting elements from our experiments and discuss some real-world implications of these findings.

The work described in this chapter has been presented in the following article:

- A. Antelmi, G. Cordasco, V. Scarano and C. Spagnuolo, "Modeling and Evaluating Epidemic Control Strategies With High-Order Temporal Networks," in IEEE Access, vol. 9, pp. 140938-140964, 2021.

10.1 Motivation

Since ancient times, different populations have adopted varying strategies to prevent and contain diseases, from isolating sick individuals to establishing a time limit to the manifestation of symptoms to magical practices [81]. The concept of modern and preventive quarantine dates back only to 1377; still today, it represents a general preventive intervention in the absence of a targeted vaccine, along with high healthcare surveillance and public information [81, 121, 148].

Generally, all healthcare policies intended to mitigate the effects of the spread of a new virus or pathogens when no vaccines or medicine are available yet are commonly referred to as Non-Pharmaceutical Interventions (NPIs) [148]. Current development of the recent pandemic highlighted to what extent the increase of human mobility and goods exchange made NPIs (e.g., lockdown and border closure) more challenging to ap-

ply to past cases in history, mainly because of their negative impact on the worldwide economy [71, 24] as well as on the psychological wellness of society [88, 184, 160, 187]. Choosing the correct control policy to adopt is a burden that governments bear as their decisions have repercussions also when the epidemic is under control. Thus, each country is responsible for adopting NPIs according to its territory's specific needs. As examined in a document published by the World Health Organization (WHO) in 2019 [148], each NPI has different effects, resource implications, and ethical considerations. To precisely study the consequences of each intervention on the region it has to be applied, the models used to simulate an epidemic propagation need to guarantee a high level of accuracy to ensure both efficacy and efficiency in evaluating the specific control policy.

In Chapter 9, we already discussed the benefits of using ABMs over EBMs to simulate human behaviors under given conditions by embedding society-specific aspects within the model [42]. Further, ABMs are usually employed as a tool for what-if analysis and perfectly fit this particular context, since estimating and evaluating the impact on society deriving from the application of NPIs is challenging, given the numerous and closely tied aspects to examine. As Nicola Perra notes in his survey on modeling NPIs to contrast the COVID-19 spreading [153], ABMs are typically used to understand better how the features of real contact networks might affect the unfolding of a virus by abstracting some societal details (such as households, schools, and workplaces) and, thus, reducing computational costs. For instance, ABMs have been used to quantify the effects of contact tracing, isolation, and vaccination campaigns. Very complex ABMs to make accurate predictions exist too (see Chapter 5).

Chapter 9 also reviews the motivations behind modeling contact networks using high-order structures, specifically hypergraphs, in epidemiological studies. In this context, hypergraphs account for community structures, infection pressure, and indirect contagion pathways (see Chapter 5). These high-order models have successfully shown their expressiveness in analyzing epidemic diffusion phenomena, and evaluating blocking strategies via such structures represents the natural evolution of this

research trend.

10.2 Modeling NPIs

NPIs are healthcare policies readily available at all times and in all countries, intended to mitigate the effects of the spread of a new pathogen or virus when no vaccines or medicine are available yet [159]. The potential impacts of NPIs on an influenza epidemic are to delay the introduction of the infection into a population, delay the height and peak of the outbreak if it has started, reduce transmission by personal protective or environmental measures, and reduce the number of infections and hence the number of severe cases [148]. In 2019, the WHO Global Influenza Programme and the WHO Collaborating Centre for Infectious Disease Epidemiology and Control (School of Public Health, University of Hong Kong) published a report providing recommendations for the use of NPIs based on a systematic review of the evidence on their effectiveness, including personal protective, environmental, social distancing, and travel-related measures [148].

In the remainder of this section, we will review personal protective, environmental, and social distancing measures and how they can be embedded into an epidemiological model based on high-order networks, ABMs, and the SIS equation-based model. Moreover, we describe how we formally enriched our modeling framework to support the evaluation of NPIs.

10.2.1 Personal Protective Measures

Examples of personal protective measures (PPMs) are hand hygiene, respiratory etiquette, and face masks. While the first two actions are a well-established and straightforward practice concerning personal daily hygiene, face masks are only conditionally recommended during a severe epidemic or a pandemic or for symptomatic individuals. PPMs can be implemented and simulated by decreasing the transmission probability of a pathogen from one individual to another when they come in contact

and changing the infectiousness probability caused by interacting with environments. PPMs are an instance of individual behavior interventions as their efficiency is defined by how many people respect them.

We embedded the adoption of PPMs into our model by reducing the epidemic spreading opportunity (transmission probabilities) for both direct and indirect contagion pathways. Specifically, we decreased the values of the parameters β_d (*agent-to-agent*), β_i (*environment-to-agent*), and β_e (*agent-to-environment*) to simulate the introduction of PPMs into the agent population; hence, leaving unchanged the three phases of the diffusion algorithm (see Section 9.2.3). It is worth noting that we lower the value of those parameters only for the agents adopting PPMs.

10.2.2 Environmental Measures

Surface and object cleaning actions are environmental measures (EMs) recommended as a public health intervention in all settings to reduce influenza transmission. As for PPMs, environmental measures can be implemented by decreasing a pathogen's transmission probability from a contaminated object to an individual. For instance, if a person sneezes on a table and, soon after, another person touches that surface, a contagion may indirectly occur from one person to another. To simulate EMs, we need to embed within the network of contacts the notion of *environment* or *location* where people can interact either with it or with other individuals. In a nutshell, the network model used to study epidemic propagation has to be location-aware. Here, hypergraphs come into play as a hyperedge can naturally model a group of people being in the same location in a given time, even though they did not have any direct contact (see Section 9.2).

Similar to PPMs, the adoption of EMs can be simulated by modifying the contagiousness probabilities β_i and β_e. This approach can reproduce either a global adoption of some regulation (e.g., cleaning all public or private spaces at every use) or more realistic settings, where only specific locations are sanitized after each use (e.g., a table in a restaurant) or at regular time intervals (e.g., trains). In our experiments, we reduced the

parameter regulating the contagiousness of an indirect contact β_i by a factor β_ℓ, with $\beta_\ell \leq \beta_i$, for all locations $\ell \in \mathcal{L}$ sanitized at given time intervals. Formally, $\beta_i \leftarrow \beta_i - \mathrm{X}_{\mathcal{L}}(\ell)\beta_\ell$, where $\mathrm{X}_{\mathcal{L}}(\ell)$ is an indicator function equal to 1 for all the locations that are continuously sanitized. If $\beta_\ell = \beta_i$, the transmission probability becomes $\beta_i = \beta_i - \beta_\ell = 0$. Thus, it causes the place to be no more infectious.

As detailed in Section 10.5.2, we simulate the cleaning of all locations restoring their status to susceptible; thus, temporarily reducing the parameter β_i to 0. It is worth noticing that locations may become contaminated again. In this case, β_i assumes its original value. The implementation of EMs required the addition of a fourth phase after the three described in Section 9.2.3.

10.2.3 Social Distancing Measures

Social distancing measures (SDMs) represent interventions on individuals' sociality and involve the population or sub-population behaviors. Examples of these measures are (i) isolation of sick individuals, (ii) quarantine of exposed individuals, (iii) contact tracing, (iv) avoiding crowding, and (v) school, workplace, and, in general, public or private structure closures.

The design of SDMs requires a different level of detail according to the policy to simulate. For instance, isolating an individual assumes to have access to the infection state of a person to know whether he/she is sick. Implementing quarantine measures adds another layer of complexity as the simulation has to account for an ill individual's contacts. Hence, correctly estimating the connections a person had is critical to an adequate estimation of the effects of an SDM. Contact tracing measures are based on the same principle: direct contacts between two individuals are stored to retrace the contact chain if needed [120, 72]. In principle, without dwelling on ethical and privacy concerns, contact tracing strategies could also exploit data deriving from indirect contacts among individuals. For instance, if a sick individual has been in a restaurant, all people in the same place must be notified as they could have touched contaminated

surfaces (e.g., bathroom or doors). Further, simulating lockdown or social distancing policies based on real-world mobility patterns highlights the need for specific location data, such as the type of the place (e.g., a cafe, a workplace, or a station).

In the following, we describe how we embedded these measures within our modeling approach.

Isolation

The word isolation indicates the separation or restriction of movement of ill individuals with an infectious disease to prevent transmission to others [74].

We embedded this policy within the behavior of each agent, able to recognize whether it is sick. At each time interval and before the execution of the three phases of the diffusion algorithm, every agent a may go in self-isolation according to a probability $\beta_{isolation}$, regulated by the following Poisson process $1 - e^{-T(a) \cdot \beta_{isolation}}$, where $T(a)$ is the infection state of the agent a. When an agent is isolated, it does not contribute to the epidemic spreading; in other words, other agents or environments cannot interact with an isolated agent. Once isolated, the agent does not exit this status until it recovers with a probability γ_a.

Quarantine

The word quarantine indicates an imposed separation or restriction of movement of individuals exposed, who may or may not be infected but are not ill, and who may become infectious to others [74].

As for the isolation measure, we embedded this policy within the behavior of the agents. At each time interval t and, specifically, during the *Agent-to-Agent* contagion phase, an agent a enters the quarantine state according to an overall probability $\beta_{quarantine}$ proportional to the number of infected agents $I_t^d(a)$ it has met during the previous time step. This scenario is regulated by the Poisson process $1 - e^{-I_t^d(a)\beta_{quarantine}}$, where the number of ill agents $I_t^d(a)$ is computed based on the simulation status of

the previous time step $(t-1)$. A quarantined agent no longer contributes to the epidemic until it exits the confinement state or recovers with a probability γ_a.

Tracing

The idea behind contact tracing applications is to rapidly identify at-risk individuals once a case has been detected [67]. However, even though such systems can substantially increase the proportion of people quarantined, it can lead to ethical issues such as leakage of information [148]. Furthermore, successful implementation relies on the availability of resources and technology [120, 72]. Generally, a design tool has to trace all agents' contacts during the simulation and identify infected agents. To simplify the evaluation of a contact tracing intervention, we introduced several abstractions in our model. Specifically, we did not consider any error in the retracement process, assuming a 100% accuracy of location data, and we did not implement any particular contact tracing protocol [3]. Further, we assume that all agents adopting the tracing software know whether the other agents they met were healthy or infectious (here, we do not explicitly model any testing protocol). In other words, we are considering a fully functioning tracing system.

Contact tracing is modeled similarly to the quarantine measure, but in this case, the tracing system is more powerful and enables tracking the number of infected individuals the agent has met during all previous steps instead of only the previous one. Put differently, we can consider this measure like an informed quarantine as the main difference between these two interventions lies in the information used by an agent to decide whether to enter the quarantine state. In this case, the agent *remembers* the contacts had in all previous steps since the tracing measure was adopted. The usefulness of distinguishing these two scenarios is also evident in a more complex compartmental model with an Exposed state. In this case, adopting a tracing technology allows the agent to keep track of all its contacts and know whether they have manifested symptoms or tested positive on the pathogen even though they have come across many simula-

tion steps before. When only quarantine measures are implemented, the agent only uses the contact information available in the previous simulation step. As a consequence, the agent may self-quarantine with a lower probability.

Formally, for each agent a using the tracing application, we introduced a list $\mathcal{N}(a)$ comprising all the agents (using the tracing application themselves) that interacted with a during the simulation. At each simulation step and before the execution of the three phases of the diffusion algorithm, every agent a, either infected or not, may decide to self-quarantine according to the Poisson process $1-e^{-I^d(a)\beta_{tracing}}$, where $I^d(a)$ is the number of infected individuals the agent has met during all previous steps. If an agent sets its state to quarantine, then the simulation proceeds as for the implementation of the quarantine measure.

Avoiding Crowding

Avoiding crowding is another example of SDM, often used with other policies to reduce influenza transmission. Avoiding crowding may have cultural or religious implications; for instance, gatherings are places to share information during a pandemic, comfort people, and reduce fear. During the SARS-CoV-2 pandemic, many countries adopted several avoiding crowding policies - some stricter than others - to contain its spread. For instance, the English government permitted only up to six people to meet [143], while the Italian government banned all sorts of gatherings [63].

Our framework implements the avoiding crowding measure as a global policy by deterministically reducing the number of agents allowed within each place. We defined a threshold α_ℓ representing the maximum number of people allowed in a location ℓ per time interval. Specifically, we only simulate the first α_ℓ agents (sorted by their check-in time) at each time step for each location ℓ. As the input data determine the scheduling policy of each agent, the newly reduced data set can be easily pre-computed without requiring modifications to the diffusion algorithm.

Location Closure

Closing public or private places is a standard measure to control an epidemic spreading by reducing possible contact between individuals and environments in a severe outbreak. This extreme measure may have substantial economic consequences and result in significant societal problems.

Typically, this intervention is implemented by selectively closing places according to a specific classification, such as schools, transportation, workplace, restaurants. We implemented this policy by stopping the simulation of all direct and indirect contacts happening within a location ℓ belonging to the set of closed places $\mathcal{L}$, starting from a time interval t. Hence, implementing such a measure required modifying all three phases of the diffusion algorithm to prevent the simulation of all contacts within the closed locations. This measure can be formally illustrated by reducing to 0 all parameters regulating either an agent or an environment's infectiousness $\beta_{(\cdot)} \leftarrow \beta_{(\cdot)} - \mathrm{X}_{\mathcal{L}}(\ell)\beta_{(\cdot)} \quad \forall \beta_{(\cdot)} \in \{\beta_i, \beta_d, \beta_e\}$, where $\mathrm{X}_{\mathcal{L}}(\ell)$ is an indicator function equal to 1 for all closed locations (i.e., $\ell \in \mathcal{L}$). The set $\mathcal{L}$ can be pre-computed according to the policy to be investigated (e.g., closing transportation or workplaces).

Exploiting the dual hypergraph

Building the dual of the agents-locations hypergraph in this modeling context means considering locations as vertices and agents as hyperedges. Although the dual hypergraph encodes the exact information of the original structure, it allows - for instance - the use of centrality measures defined for vertices and clustering algorithms to identify particularly important venues or specific groups of locations as subjects of focused interventions. In practice, this modeling choice may be helpful in the design of lockdown poli-

cies to decide which venues may be worth closing or limiting their capacity.

10.3 State-of-the-art Comparison

Typically, every ABM responds to particular needs, simulating an environment under given conditions and focusing on specific aspects of the real world to model with different levels of abstraction. As a consequence, directly comparing two ABMs is practically impossible. Further, ABMs are usually validated with external data, and their performance (in terms of correctly modeling the real-world dynamics) is compared against this data.

Based on the above consideration, the first key difference between our work and related literature lies in how we model (i) the contact network and (i) the contagion dynamics via direct and indirect contacts (see Section 9.3). In this chapter, we exploit the TVH framework introduced in Chapter 9 to model direct and indirect transmission dynamics of disease spreading. Similar to (temporal) graphs, TVHs abstract and formalize contact among agents simulated with an ABM, but they add information about where the contact is happening. Hence, such structures allow to formally analyze diffusion mechanisms while accounting for group interactions and indirect contagion processes via contaminated locations. We further embedded the formal definition of NPIs within the TVH framework. The main objective of our study represents the second significant difference with the discussed literature as those works specifically focus on modeling NPIs to prevent the COVID-19 diffusion. In contrast, we generally study the diffusion dynamics of a pathogen that can also spread indirectly via human-to-environment contact.

A work that needs to be cited in this section is the study of Bouchnita et al. [45], proposing a multi-scale ABM which considers both direct and indirect transmission mechanisms. The model does not explicitly consider the notion of location as agents move on a grid and simulates

indirect contagion based on the normalized concentration of deposited SARS-CoV-2 on hard surfaces, the averaged rate of SARS-CoV-2 secretion by contagious agents, and the decay rate of the virus. For each agent, indirect transmission can occur only once every day at a random moment. As discussed in Section 9.2.2, the indirect diffusion mechanism described by Bouchnita et al. is profoundly different from the process described in Chapter 9 as hyperedges encode places where agents can meet and may become contaminated according to Poisson processes proportional to the number of infected nodes within that hyperedge.

10.4 Experiment Setting

This section describes the experimental setup of the simulations, recalling the underlying epidemiological assumptions and detailing the simulation parameters. It further defines how each NPI is evaluated and finally describes the data sets used. The experiment is designed to evaluate the impact of NPIs not only in terms of reducing the fraction of infected but also considering the cost of applying each measure.

Code Availability

The model and the experiments are implemented in Julia, exploiting the library `SimpleHypergraphs.jl` (see Chapter 7). The code is open-source and available at the following GitHub public repository: `https://github.com/alessant/HGEpidemics`.

10.4.1 Assumptions

The experiment setting of this chapter inherits all the assumptions described in Section 9.4.1. We briefly recall them in this section for readability purposes. We assume that: (*i*) all infected are asymptomatic, (*ii*)

there is no incubation phase, and (*iii*) the parameters regulating the epidemic are fixed. In these experiments, we further assume that each NPI is perfectly applied.

10.4.2 NPIs Evaluation

To implement each NPI, we modified the diffusion algorithm described in Section 9.2.3 according to the policy to simulate. The parameters and framework enrichment for each technique are detailed in Section 10.2. The impact of an NPI or a combination of NPIs is compared against an unmitigated scenario in which no interventions are implemented. We evaluated the effectiveness of each NPI (or a combination of them) according to two parameters: the reduction of the final fraction of infected agents and how much the intervention affects the population. To quantify each action's impact and compare the results across all data sets, we operatively defined two domain-agnostic notions of *damage*. Specifically, for each agent, we determined the fraction of agents it was unable to meet because of the adopted interventions. We then defined the overall social damage of the intervention $\mathcal{D}_a$ as the average over the whole population. Similarly, for each agent, we computed the fraction of the locations it was not able to visit, due to the adopted interventions, and then evaluated the overall commuting damage $\mathcal{D}_l$ as the average over the population.

10.4.3 Simulation Parameters

In the unmitigated (baseline) scenario or when no specified otherwise, we set the epidemic parameters as follows: $\beta_d = 0.57$, $\beta_i = 0.29$, $\beta_e = 0.29$, $\gamma_e = 0.18$, $\gamma_a = 0.024$, and $c = 5$. To have reasonable parameter values, we based their choice on the mathematical model *SIDHARTE* proposed by Giordano et al. [82] for the SARS-CoV-2 pandemic. We fixed the values of $\Delta = 4$ (4 hours) for an indirect contact to happen, to mimic the resistance of the COVID-19 on surfaces [149], and $\delta = 15$ (15 minutes), based on the Immuni mobile tracing app that considers a direct contact happening in a time window of 15 minutes [86]. Thus, in our model, an indirect contact may occur if two people have been in the same place within 4

hours of difference, while a direct contact may happen if two people have been in the same place within 15 minutes. It is worth underlining that our goal is not to simulate the diffusion of the SARS-CoV-2 virus, but more generally, the diffusion of an epidemic that can spread even indirectly. In this setting, the average time an agent is infected is $42 \cdot \Delta$ intervals (7 days), while for a location is $6 \cdot \Delta$ intervals (1 day). Table 10.1 lists all the simulation parameters used to control the epidemic spreading and the introduction of NPIs.

We generated the population mobility pattern according to the three data sets presented in Section 10.4.4. The simulations last 185 steps while the interventions (if any) are introduced at the $100th$ step. As the fraction of infected results from probabilistic processes, we ran each simulation scenario 80 times, and considering the averaged value as a result. The simulation is initialized with a single infected agent, the patient zero.

It should be emphasized that, in an SIS model, recovered individuals become susceptible again without gaining any immunity against the pathogen. In other words, there is no memory of past infections. This is the typical spreading model of infections like the common cold and influenza, which do not confer any long-lasting immunity. In contrast, in the SIR model, all agents will eventually recover and not contract the infection anymore. In other terms, in the SIR model, the outbreak will ultimately drop out. Based on this consideration, and as we are interested in evaluating control actions in the worst scenario, we analyze each intervention under the SIS model.

10.4.4 Data Sets

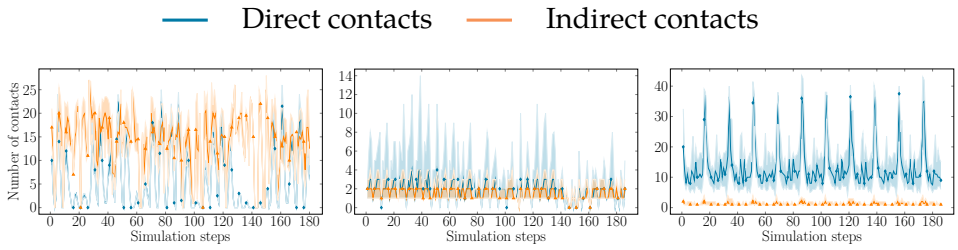
In the ABM model we developed, each agent moves between geo-locations over time and comes in contact, via direct or indirect pathways, with other agents and different environments (geo-locations). To model individuals (agents) mobility patterns, we adopted three data sets describing human interactions at different scale: from (*i*) a location-aware sensing infrastructure (BLEBeacon data set [166]), (*ii*) a metropolitan area scenario (Foursquare data set [190]), and (*iii*) a virtual society scenario (Game

Table 10.1: Simulation parameters. For each parameter, the table reports: which aspect of the simulation the parameter regulates, a short description, its domain, and its value in the simulation (*variable* indicates that the parameter changes according to the specific experiment).

Type	Parameter	Description	Domain	Value
High-order epidemic spreading	β_d	Probability that an agent is infected by another agent via direct contact.	$[0, 1]$	0.57
	β_i	Probability that an agent is infected via indirect contact due to a location.	$[0, 1]$	0.29
	β_e	Probability that a location is infected by an agent.	$[0, 1]$	0.29
	γ_a	Probability that an agent spontaneously recovers.	$[0, 1]$	0.024
	γ_e	Probability that a location is sanitized.	$[0, 1]$	0.18
	c	Number of contacts in *Agent-to-Environment*.	$\mathbb{N}$	5
PPMs	α_p	Fraction of agents using PPMs.	$[0, 1]$	Variable
	ppm_β_d	Probability that an agent is infected by another agent via direct contact when PPMs are in use.	$[0, 1]$	0.1
	ppm_β_i	Probability that an agent is infected via indirect contact due to a location when PPMs are in use.	$[0, 1]$	0.05
	ppm_β_e	Probability that a location is infected by an agent when PPMs are in use.	$[0, 1]$	0.05
EMs	`sanitize`	Whether locations are regularly sanitized.	{`true`, `false`}	Variable
SDMs	$\beta_{isolation}$	Probability that an agent enters the isolation state.	$[0, 1]$	Variable
	$\beta_{quarantine}$	Probability that an agent enters the quarantine state.	$[0, 1]$	Variable
	α_e	Fraction of location closed during the lockdown.	$[0, 1]$	Variable
	α_i	Fraction of agent using a tracing application.	$[0, 1]$	Variable
	$\beta_{tracing}$	Probability that an agent enters the quarantine state if it adopts tracing technologies.	$[0, 1]$	0.6
	`avoiding crowding`	Whether avoiding crowding measures are applied.	{`true`, `false`}	Variable

of Thrones data set [103]) - all described in Chapter 6.

Figure 10.1 shows the median number of direct and indirect contacts across all data sets, considering $\Delta = 4$ hours and $\delta = 15$ minutes. For each value, we also report the 25% and 75% quantiles. In more detail, as direct contacts, we evaluated for each agent the number of different other agents it met; while, as indirect contacts, the number of different locations the agent has been. These plots depict an *agent-centered* vision of the data, in the sense that we can grasp, on average, how many different ways an agent may be infected by counting the number of direct contacts and locations visited (indirect contacts). Figure 10.1 reveals the diverse nature of the data sets, which results in distinct contact patterns. The BLEBeacon data set refers to high-granular small-scale check-ins happening within a building. Based on that, it is reasonable to think that an individual tends to meet always the same people, but - at the same time - they are free to walk in the structure. Figure 10.1a details this pattern where the number of indirect contacts is generally higher than the number of direct contacts. We can observe a completely different picture in Figure 10.1c, related to the Game of Thrones (GoT) data set, where the number of direct contacts is significantly higher than the number of indirect contacts. Once again, this pattern is due to the constitution of the data as the majority of the GoT characters tend not to move across many different locations, but they still have many direct contacts (e.g., council or battle scenes). Figure 10.1b refers to the real-world large-scale check-in data set from the social plat-



(a) BLEBeacon. (b) Foursquare. (c) GoT.

Figure 10.1: Distribution of direct and indirect contacts ($\Delta = 4$ hrs and $\delta = 15$ min.).

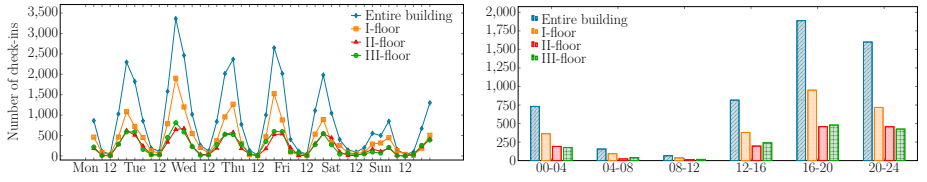
form Foursquare. In this case, no class of contacts strongly prevails on the other. In contrast with the other two data sets, we can notice a consistent variance in the number of locations visited by the users. As already discussed in Chapter 9 and other several studies [189, 84, 19], this behavior is typical of online social networks, where a minority of the users accounts for the most content. A detailed description of each data set follows.

The BLEBeacon Data Set

The BLEBeacon data set [166] is a collection of Bluetooth Low Energy (BLE) advertisement packets/traces generated from BLE beacons carried by people following their daily routine inside a university building for a whole month. The data set contains $153,868$ check-ins of 62 users and 31 locations. Figure 10.2 provides the check-ins distribution aggregated over a week (see Figure 10.2a) and over a day (see Figure 10.2b).

The Foursquare Data Set

The Foursquare social network data set, introduced by Yang et al. in [190], is a collection of check-ins originated from the city of Tokyo and crawled from 12 April 2012 to 16 February 2013. The data set contains $573,703$ check-ins of $2,293$ users and $61,858$ locations (such as restaurants, cinemas, sports, and so on). In Chapter 9, we analyzed the sensitivity of the proposed TVH model to the simulation's parameters over the most



(a) Aggregated weekly check-ins.

(b) Aggregated daily check-ins.

Figure 10.2: Aggregated weekly (on the left) and daily (on the right) check-ins distribution of the overall BLEBeacon data set referring to 30 days of people daily routine inside a university building.

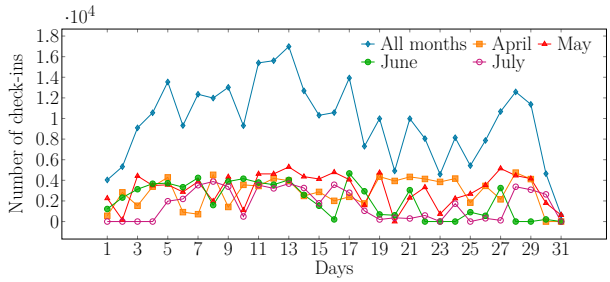
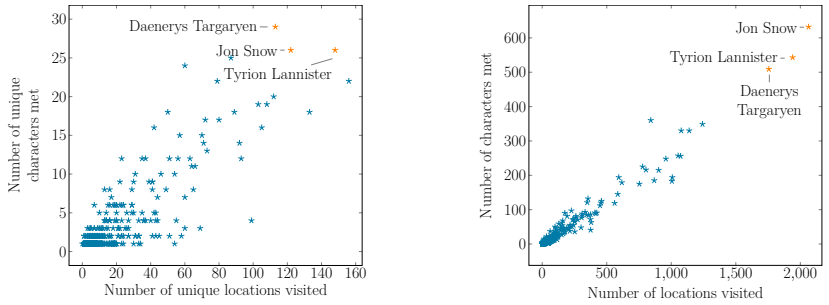


Figure 10.3: Daily check-in distribution for the Foursquare data set.

crowded month (May 2012) of this data. Having fixed $\Delta = 4$ and $\delta = 15$, our analysis revealed a peak in the number of infections near the 30% of the overall population. Towards this, it is worth stressing that the application of a single or a combination of non-pharmaceutical measures only makes sense in a dangerous scenario, in which the epidemic spreading is hard to control. Thus, to increase the probability of having a more virulent pathogen spreading and overcome the sparsity nature of the data set, we merged the check-ins happening from April 1^{st}, 2012 up to August 1^{st}, 2012 into a single month, obtaining $2,147$ users and $41,519$ locations. Figure 10.3 shows the number of daily check-ins over April-July and the final amount used for the simulation.

The GoT Data Set

Starting from scripting data [103] of the whole 8 seasons of the *Game of Thrones* (GoT) HBO TV series, we developed a check-in data set based on the mobility patterns of the characters' series. As episodes are chronologically ordered, but no real date is available, we set a virtual clock to January 1^{st}, updating it according to each scene's duration. As each episode has an average duration of around 1 hour, we obtained a data set spanning over 70 hours. As for the Foursquare data set, to better estimate the effect of NPIs in a virulent scenario on a longer temporal interval, we finally concatenated the obtained check-ins until having data covering a total of one month, with 577 characters and 111 locations. Figure 10.4 depicts



(a) *Unique*. (b) *Complete*.

Figure 10.4: GoT characters (nodes) and environments (hyperedges) distribution, considering an agent's contact history without (10.4a) and with (10.4b) repetitions.

an analysis of the characters' mobility patterns over the first 70 hours. Specifically, we examined a possible correlation between the number of unique (without repetition) people met and unique locations visited by each GoT character (see Figure 10.4a). As shown, the characters that had contacts with a higher number of people tended to have traveled across many places. This behavior becomes even more evident in Figure 10.4b, where we considered all characters met and locations visited.

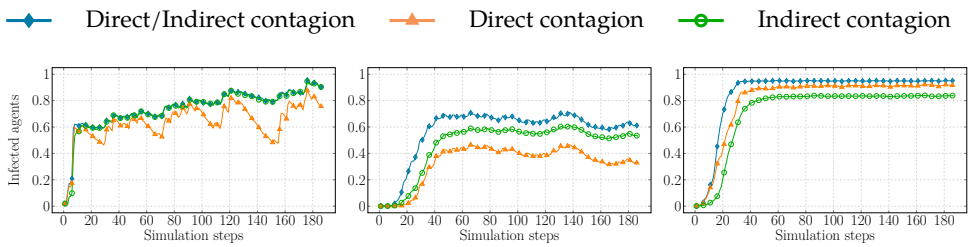
10.5 Sensitivity Analysis

This section describes the experiments we carried out to validate the proposed model and verify the correctness of its implementation. In Chapter 9, we analyzed the sensitivity of the TVH model to the epidemic parameters and different discretization of the time intervals when either direct or indirect contacts may happen. In this section, we focus on evaluating the model's sensitivity regarding the parameters regulating each NPI; thus, studying how perturbations in the input modify the model output.

To analyze the impact of reducing the fraction of infected and the damage brought by each NPI, we defined several scenarios in which we ap-

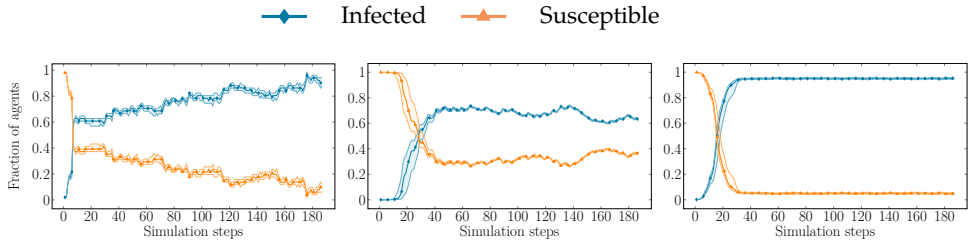
plied a single intervention. In this way, we not only evaluate the sensitivity of the model to the specific NPI, but we also verify the implementation of each intervention and ensure that each logical component of the model behaves as intended. In the following sections, we describe each validation scenario, comparing the fraction of infected agents at (*i*) the peak, (*ii*) the lowest value reached by the infection, and (*iii*) the end of each simulation against the unmitigated scenario. We further report the impact of the intervention in terms of damage as defined in Section 10.4.2. For each scenario, we also discuss some implications of using the given measure and its efficacy.

Figure 10.5 shows the epidemic diffusion patterns in an unmitigated scenario, considering contagions due to (*i*) direct and indirect contacts, (*ii*) direct contacts, and (*iii*) indirect contacts. In this case, the three plots report an *epidemic-centered* vision, in which we observe the fraction of infected due to either direct, indirect (infected locations), or both contact types. As expected, these patterns are a consequence of the contacts happening in the data. In fact, if we compare Figure 10.1 with Figure 10.5, we can note that indirect contacts cause higher peaks in the fraction of infected for the BLE data set since agents visits at least 10 locations on average while having a lower mean of direct contacts (see Figure 10.5a). On the contrary, direct contacts trigger more contagions than indirect contacts for the GoT data set, as suggested by the respective number of the two types of interactions (see Figure 10.5c). On the other hand, the contagion patterns in the Foursquare data set never reach the peak of 80% of infected (see Figure 10.5b). Once again, this outcome was expected given the high sparsity of the data. Here, indirect contagions prevail as we have more information about the places each user visits rather than the other people they meet. Further, it is worth noting that even though each agent visits only a few locations on average (see Figure 10.1b), their check-ins tend to refer to common places like transportation or general entertainment (for instance, the 41 most crowded locations in the data set refer to the transportation system). Hence, indirect contacts drive the epidemic diffusion as one location can potentially infect many agents. Consequently, these



(a) BLEBeacon. (b) Foursquare. (c) GoT.

Figure 10.5: Pathogen diffusion in an unmitigated scenario, considering contagions due to i) both direct and indirect contacts, ii) direct contacts, and iii) indirect contacts.



(a) BLEBeacon. (b) Foursquare. (c) GoT.

Figure 10.6: Epidemic propagation behavior under the SIS model.

locations have the potential to spread the epidemic across a considerable number of agents.

Figure 10.6 presents the averaged curves of the fraction of infected and susceptible agents obtained from the simulation replications. They behave similarly to the standard SIS model, dominated by the decline of the Susceptible population and the increase of the Infected population. Dark lines in the figure represent the median of 80 simulation replications, and the shaded areas represent 25 and 75 quantiles. These curves appear more rugged than those resulting from an EBM due to the heterogeneous mixing facilitated by the contact network.

10.5.1 Scenario 1: Using PPMs

Experimental setting

As described in Section 10.2.1, we modeled the introduction of PPMs by reducing the infection probability of each agent adopting these measures. Specifically, we reduced the direct and indirect contagious transmission probability by about 80%. We set the new direct and indirect contagion probability as follows: $ppm\_\beta_d = 0.1$, $ppm\_\beta_e = 0.05$ and $ppm\_\beta_i = 0.05$ (see Table 9.2). To analyze the impact of applying PPMs, we varied the fraction α_p of agents using the measures, testing increasing values of the parameter from 0 to 1 with step size 0.05. Clearly the scenario with $\alpha_p = 0.0$ corresponds to the unmitigated scenario.

Discussion

Table 10.2 reports the results for this scenario, considering α_p equal to 0.0, $.25$, $.50$, $.75$, and 1.0. First, we need to note that the application of such intervention does not cause any social damage ($\mathcal{D}_a$) nor commuting damage ($\mathcal{D}_l$). In fact, adopting PPMs like face masks and hand hygiene does not prevent an agent from meeting other agents or freely moving across locations. Second, increasing the fraction of agents adopting PPMs causes a decrease in the fraction of infected, suggesting how the infection propagation is susceptible to PPMs usage by a growing number of individuals. However, at least 75% of the population have to use PPMs to visibly decrease the fraction of infected both at the end of the simulation and in the lowest peak of the infection in all data sets. As expected from real-world events and the vast amount of current literature [153], only applying PPMs cannot notably reduce the spreading of a pathogen, but - even in small percentages - can help lower the number of infections.

10.5.2 Scenario 2: Using EMs

Experimental setting

As introduced in Section 10.2.2, our modeling framework allows by design adopting EMs by manipulating the infection probability due to indi-

Table 10.2: Scenario 1: Using PPMs. Fraction of infected agents (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}) when α_p agents use PPMs. Each value is followed by the standard deviation. In all scenarios, the damage of the intervention is 0.0.

Data set	α_p	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$
BLEBeacon	0.00	$0.93_{\pm 0.04}$	$0.95_{\pm 0.03}$	$0.76_{\pm 0.04}$
	0.25	$0.91_{\pm 0.04}$	$0.93_{\pm 0.03}$	$0.74_{\pm 0.04}$
	0.50	$0.90_{\pm 0.04}$	$0.92_{\pm 0.03}$	$0.69_{\pm 0.05}$
	0.75	$0.88_{\pm 0.04}$	$0.92_{\pm 0.04}$	$0.64_{\pm 0.06}$
	1.00	$0.84_{\pm 0.05}$	$0.88_{\pm 0.04}$	$0.56_{\pm 0.07}$
Foursquare	0.00	$0.60_{\pm 0.14}$	$0.70_{\pm 0.16}$	$0.57_{\pm 0.13}$
	0.25	$0.54_{\pm 0.12}$	$0.65_{\pm 0.15}$	$0.52_{\pm 0.12}$
	0.50	$0.47_{\pm 0.12}$	$0.64_{\pm 0.17}$	$0.46_{\pm 0.12}$
	0.75	$0.40_{\pm 0.09}$	$0.65_{\pm 0.15}$	$0.40_{\pm 0.09}$
	1.00	$0.31_{\pm 0.06}$	$0.65_{\pm 0.13}$	$0.31_{\pm 0.06}$
GoT	0.00	$0.95_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.95_{\pm 0.01}$
	0.25	$0.93_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.93_{\pm 0.01}$
	0.50	$0.91_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.90_{\pm 0.01}$
	0.75	$0.88_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.88_{\pm 0.01}$
	1.00	$0.85_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.85_{\pm 0.02}$

rect contacts. In this validation scenario, we simulate the cleaning of all locations restoring their status to *healthy* at the end of the most crowded time intervals, i.e., 12:00-16:00, 16:00-20:00, and 20:00-24:00.

Discussion

Table 10.3 presents the results of this experiment. In this case, we analyze the epidemic spreading under two different perspectives. We report the fraction of infected (*i*) considering both direct and indirect contagious pathways and (*ii*) focusing only on the contribution made by indirect con-

Table 10.3: Scenario 2: Using EMs. Fraction of infected (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}). Each value is followed by the standard deviation. The table also reports whether contagions are only due to indirect contacts or to both direct and indirect contacts (*Type of contagion*) and whether sanitization procedures are in place (*sanitize*). In all scenarios, the damage of the intervention is 0.0.

Data set	Type of contagion	sanitize	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$
BLEBeacon	Direct/Indirect	false	$0.94_{\pm 0.04}$	$0.95_{\pm 0.03}$	$0.76_{\pm 0.04}$
		true	$0.92_{\pm 0.04}$	$0.94_{\pm 0.04}$	$0.73_{\pm 0.05}$
	Indirect	false	$0.93_{\pm 0.04}$	$0.94_{\pm 0.03}$	$0.77_{\pm 0.04}$
		true	$0.80_{\pm 0.05}$	$0.84_{\pm 0.05}$	$0.70_{\pm 0.05}$
Foursquare	Direct/Indirect	false	$0.61_{\pm 0.10}$	$0.72_{\pm 0.12}$	$0.59_{\pm 0.10}$
		true	$0.56_{\pm 0.13}$	$0.69_{\pm 0.16}$	$0.53_{\pm 0.12}$
	Indirect	false	$0.55_{\pm 0.13}$	$0.64_{\pm 0.15}$	$0.53_{\pm 0.12}$
		true	$0.49_{\pm 0.13}$	$0.59_{\pm 0.15}$	$0.46_{\pm 0.12}$
GoT	Direct/Indirect	false	$0.95_{\pm 0.01}$	$0.96_{\pm 0.01}$	$0.95_{\pm 0.01}$
		true	$0.94_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.93_{\pm 0.01}$
	Indirect	false	$0.84_{\pm 0.19}$	$0.84_{\pm 0.19}$	$0.83_{\pm 0.19}$
		true	$0.76_{\pm 0.18}$	$0.84_{\pm 0.19}$	$0.74_{\pm 0.17}$

tacts. Even though we can observe poor results in the application of cleaning procedures in the first case, we can note that the introduction of EMs has some effect in reducing the fraction of infected due to indirect contacts. This result is not surprising as agents are still free to move and propagate the epidemic making the locations infected again. As in the case of the adoption of PPMs, we have neither social nor commuting damage as there are no constraints on the mobility of the agents. Given the limited cost of EMs, these interventions can be considered good practice in real scenarios as they effectively lower the number of infected cases.

10.5.3 Scenario 3: Using SDMs - Isolation

Experimental setting

In this validation scenario, we analyze the sensitivity of the model to the implementation of isolation measures (see §Isolation in Section 10.2.3). Specifically, we studied how changes in the parameter $\beta_{isolation}$, representing the willingness of an agent to enter the isolation state if infected, reflect on the number of spreaders. We ranged the parameter in the interval $[0, 1]$ with step size 0.05. Note that the scenario with $\beta_{isolation} = 0.0$ corresponds to the unmitigated scenario when no measures are being applied. Modifying the parameter $\beta_{isolation}$ means regulating when an infected agent enters the isolation state (the higher, the sooner). Further, the probability of entering the isolation state is also proportional to the number of infected agents met (the higher the number, the higher the probability). The time spent by each agent in this state is strictly dependent on the recovery probability as the agent may exit the isolation only if it becomes susceptible again. Having set the simulation parameters as described in Table 9.2, the upper bound to the time an agent remains in isolation is 7 days on average (see Section 9.4.2).

Discussion

Table 10.4 shows the experiment results up to $\beta_{isolation} = 0.5$ as the fraction of infected drastically drops out even for small parameter values. In this case, inf_{last}, inf_{peak}, and inf_{lower} do not report the total fraction of infected, but the fraction of agents that can still spread the infection, i.e., the agents that are sick but not isolated. Increasing the value of $\beta_{isolation}$ has a sensible impact on the overall spreading since - as we can see from column $isolated_{last}$ - a conspicuous part of the agents will eventually enter the isolation state, and as a such, they will no more spread the pathogen. That results in a minimum number of effective spreaders. Another interesting outcome is that the fraction of isolated agents tends to decrease for higher values of $\beta_{isolation}$. This behavior is well-explained by the overall epidemic trend: as the fraction of infected agents decreases, the probabil-

Table 10.4: Scenario 3: Using SDMs - Isolation. Fraction of infected but not isolated agents (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}). The column $isolated_{last}$ reports the average fraction of isolated agents at the end of the simulation. $\mathcal{D}_a$ and $\mathcal{D}_l$ represent the social and commuting damage of the intervention. Each value is followed by the standard deviation.

Data set	$\beta_{isolation}$	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$	$\text{isolated}_{\text{last}}$	$\mathcal{D}_a$	$\mathcal{D}_l$
BLEBeacon	0.00	0.94±0.03	0.95±0.03	0.77±0.04	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.35±0.07	0.80±0.03	0.22±0.07	0.58±0.07	0.34±0.05	0.14±0.04
	0.10	0.22±0.07	0.79±0.04	0.08±0.04	0.68±0.09	0.46±0.04	0.22±0.04
	0.20	0.09±0.07	0.79±0.04	0.03±0.03	0.66±0.23	0.52±0.09	0.26±0.05
	0.30	0.04±0.05	0.79±0.03	0.01±0.03	0.55±0.28	0.49±0.12	0.26±0.06
	0.40	0.03±0.04	0.80±0.03	0.01±0.02	0.54±0.30	0.50±0.13	0.27±0.06
	0.50	0.02±0.03	0.79±0.03	0.00±0.01	0.47±0.28	0.47±0.13	0.25±0.06
Foursquare	0.00	0.61±0.10	0.72±0.11	0.59±0.10	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.16±0.02	0.67±0.08	0.14±0.02	0.36±0.04	0.41±0.05	0.26±0.03
	0.10	0.08±0.02	0.66±0.13	0.05±0.01	0.37±0.07	0.51±0.10	0.32±0.06
	0.20	0.04±0.01	0.63±0.18	0.02±0.01	0.34±0.10	0.55±0.16	0.33±0.10
	0.30	0.02±0.01	0.66±0.13	0.01±0.00	0.34±0.07	0.58±0.12	0.35±0.07
	0.40	0.02±0.00	0.66±0.11	0.01±0.00	0.33±0.06	0.58±0.10	0.35±0.06
	0.50	0.01±0.01	0.66±0.13	0.00±0.00	0.30±0.07	0.57±0.11	0.35±0.07
GoT	0.00	0.95±0.01	0.96±0.01	0.95±0.01	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.31±0.02	0.95±0.01	0.31±0.02	0.61±0.02	0.18±0.02	0.02±0.00
	0.10	0.18±0.02	0.95±0.01	0.17±0.02	0.70±0.02	0.28±0.02	0.04±0.01
	0.20	0.10±0.01	0.95±0.01	0.10±0.01	0.72±0.02	0.36±0.02	0.07±0.01
	0.30	0.07±0.01	0.95±0.01	0.06±0.01	0.71±0.02	0.40±0.02	0.08±0.01
	0.40	0.05±0.01	0.95±0.01	0.04±0.01	0.69±0.04	0.41±0.02	0.09±0.01
	0.50	0.04±0.01	0.95±0.01	0.03±0.01	0.67±0.04	0.42±0.02	0.10±0.01

ity of being infected declines, and, consequently, the need to isolate single agents.

In a real scenario, the effectiveness of such intervention is strictly dependent on the easiness and the availability of tools to test whether a person is infected. Nevertheless, this simple experiment emphasizes the critical importance of effectively identifying contagious individuals to decrease the epidemic curve.

As expected, both damages $\mathcal{D}_a$ and $\mathcal{D}_l$ reflect that at least 30% up to 72% of the agents are isolated (see column $isolated_{last}$), preventing them from seeing other agents or visiting locations. Fixing $\beta_{isolation} = 0.5$, agents lose on average 48% of their contacts and around 23% of the locations. The social damage $\mathcal{D}_a$ assumes lower values for the GoT data set. Once again, we have to recall the nature of the data set: most of the GoT characters meet a reduced number of other characters, thus explaining the lower values. A similar comment holds for the commuting damage $\mathcal{D}_l$.

10.5.4 Scenario 4: Using SDMs - Quarantine

Experimental setting

Similar to the previous scenario, in this experiment, we study the sensitivity of the model to the implementation of quarantine measures regulated by the parameter $\beta_{quarantine}$. This parameter represents the willingness of an agent to enter the quarantine state, and it is proportional to the number of infected agents met. As for isolation measures, modifying the parameter $\beta_{quarantine}$ means regulating when an infected agent enters the quarantine state (the higher, the sooner). The time spent by each agent in this state is strictly dependent on (*i*) whether the agent is infected and (*ii*) the recovery probability of the agents. If an agent has been quarantined even though susceptible, it will immediately exit the quarantine status in the next simulation step. Otherwise, the same rules applied for exiting the isolation state hold. It is worth recalling that our experiments simulate an SIS compartmental epidemic model, where there is no latency between the infection and the actual manifest of symptoms. The absence of an Exposed state translates into safely assuming that if an agent gets infected in a time step t, it will enter the Infected state in the following time step $(t + 1)$. Hence, an agent may leave the quarantine state after a time window Δ (4 hours) only if it is susceptible; otherwise, it will remain quarantined until it heals. Clearly, in a compartmental model including the Exposed state, the quarantine should last more than the time required by the symptoms to become evident or to have a test result.

We ranged the parameter in the interval $[0, 1]$ with step size 0.05. Note that the scenario with $\beta_{quarantine} = 0.0$ corresponds to the unmitigated scenario when no measures are being applied.

Discussion

Table 10.5 reports the results of this scenario ranging the parameter $\beta_{quarantine} \in \{0.00, 0.05, 0.10, 0.20, 0.60, 0.90\}$. Also in this case, inf_{last}, inf_{peak}, and inf_{lower} do not report the total fraction of infected, but the fraction of agents that can still spread the infection, i.e., the agents that are sick but

Table 10.5: Scenario 4: Using SDMs - Quarantine. Fraction of infected but not quarantined agents (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}). The column $quarantined_{last}$ reports the average fraction of quarantined agents at the end of the simulation. $\mathcal{D}_a$ and $\mathcal{D}_l$ represent the social and commuting damage of the intervention. Each value is followed by the standard deviation.

Data set	$\beta_{quar.}$	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$	$\text{quarantined}_{\text{last}}$	$\mathcal{D}_a$	$\mathcal{D}_l$
BLEBeacon	0.00	0.93±0.03	0.96±0.02	0.77±0.04	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.71±0.06	0.80±0.03	0.69±0.05	0.22±0.05	0.08±0.03	0.02±0.01
	0.10	0.64±0.06	0.80±0.03	0.64±0.05	0.29±0.05	0.11±0.03	0.02±0.01
	0.20	0.59±0.07	0.79±0.04	0.59±0.07	0.34±0.06	0.14±0.04	0.03±0.01
	0.60	0.55±0.06	0.79±0.03	0.55±0.06	0.37±0.06	0.15±0.03	0.03±0.01
	0.90	0.53±0.07	0.80±0.04	0.53±0.07	0.39±0.06	0.16±0.04	0.03±0.01
Foursquare	0.00	0.59±0.15	0.69±0.18	0.56±0.15	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.53±0.17	0.63±0.20	0.51±0.16	0.03±0.01	0.04±0.01	0.02±0.01
	0.10	0.50±0.16	0.62±0.19	0.49±0.15	0.05±0.02	0.07±0.02	0.03±0.01
	0.20	0.48±0.11	0.65±0.15	0.47±0.11	0.09±0.02	0.11±0.03	0.05±0.01
	0.60	0.38±0.10	0.64±0.17	0.38±0.10	0.15±0.04	0.18±0.05	0.09±0.02
	0.90	0.35±0.09	0.64±0.17	0.35±0.08	0.18±0.04	0.20±0.05	0.11±0.02
GoT	0.00	0.95±0.01	0.96±0.01	0.95±0.01	0.00±0.00	0.00±0.00	0.00±0.00
	0.05	0.79±0.02	0.95±0.01	0.79±0.02	0.14±0.01	0.05±0.01	0.00±0.00
	0.10	0.69±0.02	0.95±0.01	0.69±0.02	0.23±0.02	0.08±0.01	0.01±0.00
	0.20	0.56±0.02	0.95±0.01	0.56±0.02	0.33±0.01	0.11±0.02	0.01±0.00
	0.60	0.34±0.02	0.95±0.01	0.34±0.02	0.48±0.01	0.16±0.02	0.07±0.00
	0.90	0.27±0.03	0.95±0.01	0.27±0.03	0.53±0.01	0.17±0.02	0.02±0.00

not quarantined. As before, increasing the value of $\beta_{quarantine}$ helps reducing the fraction of infected (see the columns inf_{lower} and inf_{last}). However, even though we can observe a drastic drop of this value, the epidemic still has a consistent probability of spreading. This behavior is due to the nature of the intervention itself: quarantining individuals is a preventive measure, and, as such, even susceptible agents may enter this state. On the contrary, isolation measures directly target and isolate infected individuals. For this reason, in the first case, we have a lower, even still considerable, efficacy in reducing the fraction of infected. Once again, in a real-world scenario, the effectiveness of such intervention closely relies on the time window during which a person may infect other people but still does manifest any symptoms and the possibility of observing asymptomatic spreaders.

In this scenario, we can note very low values for $\mathcal{D}_a$ and $\mathcal{D}_l$ even when $\beta_{quarantine} = 0.9$. This outcome is well explained by how the intervention works. As already discussed, both susceptible and infected agents may be quarantined. When a susceptible enters the quarantine, it will exit the state in the following iteration; thus, the intervention causes negligible damage to those agents, still protecting them from getting the infection - even if for a small time window. Combining these two elements (quarantining infected and protecting susceptible agents) ensures a reasonable trade-off between the efficacy of the intervention in reducing the number of infected and the damage brought to the population.

10.5.5 Scenario 5: Using SDMs - Tracing Contacts

Experimental setting

In this scenario, we evaluate the sensitivity of the model to the introduction of a tracing application as a control strategy to inform each agent whether and how many infected it has met in the previous simulation steps. The parameter α_i, varying in the range $[0, 1]$ with step size 0.05, regulates the fraction of the population adopting the tracing technology. The scenario with $\alpha_i = 0.0$ corresponds to the unmitigated scenario when

no measures are being applied. The parameter $\beta_{tracing}$ controls the probability that an agent enters the quarantine state, based on the number of infected the agent knows it has been in contact. We fixed $\beta_{tracing} = 0.6$ to model that if an agent uses a tracing application, it will be more likely to enter the quarantine state if needed. As expected, we traced the contacts between individuals only if both agents were using the tracing measure.

Discussion

Table 10.6 shows the results for $\alpha_i \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$. In line with the previously simulated interventions, the more agents adopt the measure, the more the intervention effectively reduces the fraction of in-

Table 10.6: Scenario 5: Tracing. Fraction of infected but not quarantined agents (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}) when $\alpha_{tracing}$ agents use a tracing application. The column $quarantined_{last}$ reports the average fraction of quarantined agents at the end of the simulation. $\mathcal{D}_a$ and $\mathcal{D}_l$ represent the social and commuting damage of the intervention. Each value is followed by the standard deviation.

Data set	$\alpha_{tracing}$	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$	$\text{quarantined}_{\text{last}}$	$\mathcal{D}_a$	$\mathcal{D}_l$
BLEBeacon	0.00	0.92±0.04	0.95±0.03	0.76±0.04	0.00±0.00	0.00±0.00	0.00±0.00
	0.25	0.85±0.05	0.88±0.04	0.76±0.05	0.07±0.03	0.03±0.02	0.00±0.00
	0.50	0.64±0.05	0.80±0.03	0.62±0.06	0.25±0.05	0.10±0.03	0.02±0.01
	0.75	0.34±0.07	0.79±0.04	0.34±0.07	0.49±0.06	0.21±0.04	0.02±0.01
	1.00	0.06±0.07	0.79±0.03	0.07±0.07	0.70±0.06	0.33±0.04	0.06±0.01
Foursquare	0.00	0.60±0.14	0.70±0.16	0.57±0.13	0.00±0.00	0.00±0.00	0.00±0.00
	0.25	0.53±0.15	0.64±0.18	0.51±0.125	0.04±0.01	0.04±0.01	0.01±0.00
	0.50	0.43±0.09	0.65±0.13	0.43±0.09	0.13±0.03	0.13±0.03	0.05±0.01
	0.75	0.28±0.07	0.64±0.17	0.28±0.07	0.22±0.06	0.23±0.06	0.08±0.02
	1.00	0.13±0.03	0.65±0.15	0.13±0.03	0.33±0.08	0.35±0.08	0.13±0.03
GoT	0.00	0.94±0.11	0.94±0.11	0.93±0.11	0.00±0.00	0.00±0.00	0.00±0.00
	0.25	0.84±0.01	0.95±0.01	0.84±0.01	0.10±0.01	0.04±0.01	0.00±0.00
	0.50	0.63±0.02	0.95±0.01	0.63±0.01	0.26±0.01	0.11±0.01	0.01±0.00
	0.75	0.37±0.07	0.95±0.01	0.37±0.02	0.46±0.02	0.18±0.01	0.02±0.00
	1.00	0.07±0.02	0.95±0.01	0.07±0.02	0.66±0.02	0.28±0.02	0.02±0.00

fected. When the whole population adopts the tracing technology, the epidemic trend drastically drops in all scenarios. Nevertheless, at least 50% of the agents must use the application to observe a significant reduction. Comparing this experiment with the quarantine scenario when $\beta_{quarantine} = 0.6$ (see Table 10.5), we can note a higher fraction of quarantined agents on average at the end of the simulation. This outcome is explained by how the different interventions work in the simulation. An agent may decide to enter the quarantine state according to both the probability $\beta_{quarantine}$ and the number of infected agents met during each interval Δ. When the agent uses the tracing application, it may enter the quarantine still based on the probability $\beta_{quarantine}$ but, this time, the number of infected agents is evaluated over the previous intervals (starting from the interval when the intervention is applied). Hence, the number of infected agents met is generally higher in this second case and, consequently, the overall probability of entering the quarantine state.

Consistently with the previous experiment on quarantining individuals (see Section 10.5.4), the social damage $\mathcal{D}_a$ and especially the commuting damage $\mathcal{D}_l$ assume low if not negligible values. The explanation for that is the same as discussed in the quarantine scenario. These results suggest the potential impact such measure can have, even though, in the real world, policymakers have to consider a plethora of constraints when implementing similar interventions, among all privacy concerns.

10.5.6 Scenario 6: Using SDMs - Avoiding Crowding

Experimental setting

In this validation scenario, we examine the sensitivity of the model when avoiding crowding measures are applied. As described in Section 10.2.3, this intervention consists in increasing the social distancing between individuals, which implicitly means reducing the number of possible direct interactions. To implement such a policy for the BLEBeacon data set, we reduced the number of agents who could access the building during a day by simulating only half of the entire population's movements. We applied

a similar approach for the Foursquare and the GoT data sets, halving the number of agents that could access a given location.

Discussion

Table 10.7 reports the results of this experiment. As for the EMs (see Section 10.5.2), we analyze the epidemic spreading reporting the number of infected (i) considering both direct and indirect contagion pathways and (ii) focusing only on the contribution made by direct contacts. Also in this scenario, when we look at the final fraction of infected due to both types of interactions, we can observe little or no impact on the epidemic spreading. On the other hand, we can note that the introduction of such a measure effectively lowers the fraction of infected due to direct contacts and the lowest peak of the infection. This outcome is somewhat expected given the nature of the intervention itself. This measure does not affect in-

Table 10.7: Scenario 6: Using SDMs - Avoiding Crowding. Fraction of infected (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}). $\mathcal{D}_a$ and $\mathcal{D}_l$ represent the social and commuting damage of the intervention. Each value is followed by the standard deviation. The table also reports whether avoiding crowding measures are applied (AC).

Data set	Type of contagion	AC	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$	$\mathcal{D}_a$	$\mathcal{D}_l$
BLEBeacon	Direct/Indirect	false	0.91±0.04	0.96±0.03	0.77±0.04	0.00±0.00	0.00±0.00
		true	0.86±0.04	0.89±0.04	0.74±0.05	0.66±0.00	0.11±0.00
	Direct	false	0.74±0.06	0.87±0.04	0.47±0.07	0.00±0.00	0.00±0.00
		true	0.55±0.06	0.70±0.05	0.35±0.06	0.66±0.00	0.11±0.00
Foursquare	Direct/Indirect	false	0.57±0.18	0.68±0.21	0.55±0.17	0.00±0.00	0.00±0.00
		true	0.46±0.15	0.61±0.20	0.44±0.15	0.84±0.00	0.12±0.00
	Direct	false	0.30±0.17	0.43±0.24	0.29±0.16	0.00±0.00	0.00±0.00
		true	0.14±0.06	0.42±0.17	0.14±0.06	0.84±0.00	0.12±0.00
GoT	Direct/Indirect	false	0.95±0.01	0.95±0.01	0.94±0.01	0.00±0.00	0.00±0.00
		true	0.92±0.01	0.95±0.01	0.92±0.01	0.40±0.00	0.01±0.00
	Direct	false	0.91±0.01	0.92±0.01	0.90±0.01	0.00±0.00	0.00±0.00
		true	0.73±0.02	0.91±0.01	0.73±0.01	0.40±0.00	0.01±0.00

teractions between people and environments; hence the pathogen is free to propagate via indirect contacts.

The intervention cost in terms of damage tends to be higher than the other scenarios described except for the isolation measure. These values are clearly explained by the fact that some agents cannot enter a location if this already contains a number of agents equal to half of its total capacity (evaluated over the unmitigated scenario).

10.5.7 Scenario 7: Using SDMs - Location Closure

Experimental setting

In this last validation scenario, we assess the model's sensitivity to the implementation of lockdown measures (see Section 10.2.3). We selected the locations to close according to two possible scenarios. In the first scenario, we randomly chose the locations; in the second configuration, we picked the most crowded places first. The parameter α_e regulates the number of locations to close. We ranged it in the interval $[0, 1]$ with step size 0.05 to simulate partial and complete closure policies. The scenario with $\alpha_e = 0.0$ corresponds to the baseline scenario when no measures are being applied.

Discussion

Table 10.8 reports the results of the application of the two tested lockdown scenarios, simulating an epidemic spreading varying the parameter $\alpha_e \in \{0.00, 0.30, 0.60, 0.90, 1.00\}$.

If we look at the final fraction of infected at the end of the simulation (column inf_{last}), we can note how such a measure has a different impact according to the nature of the data set. For instance, we need to deny access to all rooms within the building described by the BLEBeacon data if we want to reduce the contagions drastically. In addition, closing the locations giving priority to crowded places does not seem to improve the performance over closing random rooms when $\alpha_e \leq 0.90$. Although surprising at first glance, the explanation for this outcome is due to the fact that the number of check-ins per location is evaluated over the whole data

Table 10.8: Scenario 7: Using SDMs - Location closure. Fraction of infected (averaged over all simulation runs) at the peak (inf_{peak}) and the lowest value (inf_{lower}) of the infection, and the end of the simulation (inf_{last}). $\mathcal{D}_a$ and $\mathcal{D}_l$ represent the social and commuting damage of the intervention. Each value is followed by the standard deviation. The table also reports the policy according to which locations are closed (*Sorting*).

Data set	α_e	Sorting	inf_{last}	inf_{peak}	$\text{inf}_{\text{lower}}$	$\mathcal{D}_a$	$\mathcal{D}_l$
BLEBeacon	0.00	-	0.93±0.04	0.95±0.03	0.76±0.04	0.00±0.00	0.00±0.00
	0.30	random	0.87±0.04	0.90±0.03	0.64±0.06	0.10±0.00	0.38±0.00
	0.60	random	0.85±0.04	0.88±0.04	0.60±0.05	0.25±0.00	0.67±0.00
	0.90	random	0.70±0.05	0.79±0.03	0.43±0.06	0.69±0.00	0.93±0.00
	0.30	most crowded	0.93±0.03	0.95±0.03	0.74±0.05	0.09±0.00	0.33±0.00
	0.60	most crowded	0.84±0.04	0.87±0.03	0.59±0.06	0.28±0.00	0.68±0.00
	0.90	most crowded	0.53±0.06	0.80±0.03	0.31±0.06	0.87±0.00	0.96±0.00
	1.00	-	0.13±0.05	0.79±0.03	0.13±0.05	1.00±0.00	1.00±0.00
Foursquare	0.00	-	0.57±0.18	0.67±0.21	0.55±0.17	0.00±0.00	0.00±0.00
	0.30	random	0.49±0.11	0.65±0.15	0.48±0.11	0.37±0.00	0.32±0.00
	0.60	random	0.39±0.10	0.64±0.17	0.39±0.10	0.57±0.00	0.60±0.00
	0.90	random	0.19±0.06	0.62±0.19	0.19±0.06	0.89±0.00	0.90±0.00
	0.30	most crowded	0.09±0.02	0.65±0.13	0.09±0.02	0.99±0.00	0.76±0.00
	0.60	most crowded	0.08±0.03	0.61±0.21	0.08±0.03	1.00±0.00	0.89±0.00
	0.90	most crowded	0.08±0.02	0.65±0.13	0.08±0.02	1.00±0.00	0.97±0.00
	1.00	-	0.08±0.02	0.65±0.13	0.08±0.02	1.00±0.00	1.00±0.00
GoT	0.00	-	0.95±0.01	0.96±0.01	0.95±0.01	0.00±0.00	0.00±0.00
	0.30	random	0.65±0.01	0.95±0.01	0.65±0.01	0.43±0.00	0.44±0.00
	0.60	random	0.51±0.01	0.95±0.01	0.51±0.01	0.62±0.00	0.66±0.00
	0.90	random	0.27±0.01	0.95±0.01	0.27±0.01	0.87±0.00	0.91±0.00
	0.30	most crowded	0.41±0.01	0.95±0.01	0.41±0.05	0.77±0.00	0.75±0.00
	0.60	most crowded	0.22±0.01	0.95±0.01	0.22±0.01	0.95±0.00	0.94±0.00
	0.90	most crowded	0.14±0.02	0.95±0.01	0.14±0.02	1.00±0.00	1.00±0.00
	1.00	-	0.12±0.01	0.95±0.01	0.12±0.01	1.00±0.00	1.00±0.00

set and not only over the portion when the intervention is applied. Thus, the most crowded locations may contain the most check-ins at the start of the simulation rather than the end, leading to the closure of less crowded rooms when the intervention is applied. On the other hand, we can ob-

serve a completely different picture for the Foursquare data set, in which closing 30% of the most crowded locations leads to a notable decrease in the fraction of infected. Once again, this outcome is due to the characteristic of the data. The majority of the check-ins of the original data set happens in a limited amount of places, like transportation or general entertainment. Hence, even closing a limited number of places brings down the epidemic. We have a similar outcome for the GoT scenario in which closing the most crowded locations generally achieves better results in lowering the fraction of infected than closing random locations. The results of these two scenarios are aligned with the measures we would expect in a real-life scenario. In fact, if we consider lockdown policies issued for the COVID-19 pandemic, we can note that they usually tended to penalize aggregation and leisure places. Clearly, we can note very high values for the commuting damage $\mathcal{D}_l$ due to the locations closed by the intervention. As a consequence, also the social damage $\mathcal{D}_a$ assumes high values when $\alpha_e \geq 0.50$ as agents cannot meet other agents in a closed place. When all locations are closed, we have the maximum damage.

10.6 Combining NPIs

In Section 10.5, we investigated the model output when a single intervention is applied. Other than evaluating the model sensitivity, we thus explored, at the same time, the effects of applying a single NPI in terms of effectively reducing the number of infected and costs required to implement the given measure. However, we expect to see the application of different interventions combined to contain the epidemic spreading in a real-world scenario. Which NPIs should be applied or how strict the measures should be implemented closely depends upon the gravity of the current situation, like the pressure on the hospitalization system. For instance, we experienced rigid lockdown policies during the first and second waves of the COVID-19 pandemic, while even the use of face masks was lifted during summer 2020.

Current literature on the topic investigates the application of NPIs

based on the complexity and heterogeneity of the data fed into the ABM. When the model is highly detailed, like in the case of France [93], Boston [8], Seattle, and New York [7], the simulation usually focuses on reproducing and improving the measures really implemented by the government. In the case of simpler models, other works manually combine several NPIs and examine their outcomes [191, 45, 167]. In this work, we tackle the issue of identifying feasible combinations of NPIs by approaching the problem under an optimization framework with two contrasting objectives - the fraction of infected and the damage brought by the intervention. The optimal solutions (in terms of NPI combinations) are thus found by exploring the model's parameter space via a multi-objective genetic algorithm. Hence, no handcrafted configuration is required. In this manner, we are able to unbiasedly explore the model's behavior without any assumption about the interaction of model parameters and their effects on the overall simulation. In the following, we will describe the method used for our experiments and the results obtained. A discussion section in which we review possible real-world implications follows.

10.6.1 Experiment Setting

In this study, we formalize the parameter space exploration of our model under the MOGA framework (see Section 5.4). Specifically, we use the algorithm NSGA-II (fast elitist non-dominated sorting genetic algorithm), a Pareto-based MOGA proposed by Deb et al. [62]. This algorithm alleviates the problems of lack of elitism, the need of sharing parameters, and high computation complexity characterizing its predecessor NSGA [169], still able to find a diverse set of solutions and converge near the actual Pareto-optimal set. We used the NSGA-II algorithm and its Julia implementation, available at the following GitHub repository[1]. We examine the multi-objective problem of balancing the use of NPIs to control an epidemic spreading and the negative impact on the overall functioning of society considering as contrasting objectives (i) reducing the final fraction of infected at the end of the simulation, (ii) the social damage $\mathcal{D}_a$, and (iii)

[1] `https://github.com/gsoleilhac/NSGAII.jl`

the commuting damage $\mathcal{D}_l$.

Each individual (i.e., a setting) is described by a 7-element vector, where each item, ranging in the interval $[0, 1]$, represents whether or not a given NPI is applied and to what extent. We considered the same measures described in Section 10.2 and analyzed in Section 10.5, namely:

- *PPMs*. Measure regulated by the parameter α_p, describing the fraction of agents using PPMs.
- *EMs*. Measure regulated by the boolean parameter *sanitize*, representing whether locations are regularly sanitized.
- *SDMs*
  - *Isolation*. Intervention regulated by the parameter $\beta_{isolation}$, representing the willingness of an agent to enter the isolation state if infected.
  - *Quarantine*. Intervention regulated by the parameter $\beta_{quarantine}$, describing the willingness of an agent to enter the quarantine state.
  - *Tracing*. Measure regulated by the parameter α_i, indicating the fraction of the population adopting tracing technologies.
  - *Avoiding crowding*. Measure regulated by the boolean parameter *avoiding crowding*, representing whether avoiding crowding measures are applied.
  - *Lockdown*. Intervention regulated by the parameter α_e, describing the fraction of locations to close. We picked the most crowded places first, given the best outcome in terms of reduction on the fraction of infected evaluated in Section 10.5.7.

The boolean parameters are considered `true` in the simulation if the value of the corresponding item in the individual is at least equal to 0.5.

We run NSGA-II for 100 generations, using a population of 100 individuals. The remaining parameters of the simulation are described in Section 10.4.

10.6.2 Results

In this section, we go through the outcomes of our analysis, first observing the Pareto front output of the algorithm NSGA-II. We then zoom into the characteristics of the solution nearest to the ideal point, trivially identified with the origin $(0, 0, 0)$, indicating that there are no infected and no damage due to the interventions. This solution is further compared with the two NPI configurations reaching the lowest number of infected and the lowest damage. We show the characteristics of each configuration in a radar chart, in which we report all interventions counterclockwise according to their damage. The color of the external circular shape (yellow-orange-red) encodes the damage of each intervention: the darker, the higher. Visually, low-damage solutions tend to pick interventions in the upper half of the circle. On the contrary, high-damage configurations tend to select higher values for interventions located in the lower half of the chart. We finally discuss how these configurations impact the epidemic spreading, and the damage paid for their application.

In more detail, every plot highlights three NPI configurations: (*i*) the configuration reaching the lowest number of infected at the end of the simulation (*Lowest-Infected*, depicted in blue), (*ii*) the configuration obtaining the lowest damage (*Lowest-Damage*, depicted in orange), and (*iii*) the configuration nearest to the optimal point $(0, 0, 0)$ according to the euclidean distance (*Nearest-to-Ideal*, depicted in green). We also added to each plot the two settings when no interventions are in place (*No-NPIs*) and when all NPIs are used to the fullest (*Full-NPIs*). All configurations come from the Pareto optimization and lies on the Pareto front analyzed. It is worth noting that the optimization algorithm explores the overall Pareto front; however, the NPI combinations that decision-makers should investigate are the solutions that guarantee the best balance between the three contrasting objectives. As a matter of fact, in this specific context is straightforward to handcraft an intervention that brings no damage (trivially selecting no NPIs or only PPMs or EMs as they provide no damage) or aims to minimize the number of infected (e.g., implementing all NPIs

available). Still, none of these interventions are truly useful in a real-world situation. The optimization algorithm can hence guide policymakers towards examining the best NPI combination according to the actual data.

Simulation parameters

It is worth noticing that each NPI configuration mentioned in this paragraph (e.g., Nearest-to-Ideal) is an individual produced by the algorithm NSGA-II. Hence, it encodes whether or not a given NPI is applied and to what extent (see Section 10.6.1). In the following, the specific values assigned to the simulation parameters regulating NPIs directly derive from particular individuals produced by NSGA-II.

The BLEBeacon Scenario

This paragraph discusses the outcome of the BLEBeacon data set, resembling a social event happening in a month timeframe in a university building (see Section 10.4.4). Figures 10.7, 10.8, and 10.9 report the results.

As we can see in Figure 10.7, the Pareto front evaluated by the algorithm NSGA-II spans over a broad spectrum of solutions, but, as observed in the introduction of this section, we will focus on the characteristics of the combination nearest to the optimal point (Nearest-to-Ideal). The first element to note is that this NPI configuration only selects policies located in the yellow-orange zone of the radar chart, causing the lowest damage (see Figure 10.8b). Among those, we can observe that all agents use PPMs ($\alpha_p = 1.0$) and that locations are regularly cleaned. We can further notice that almost all agents exploit tracing technologies ($\alpha_i = 0.96$). This choice translates into an overall significant probability for agents to self-quarantine themselves. It should not surprise that this particular configuration does not implement any basic quarantine mea-

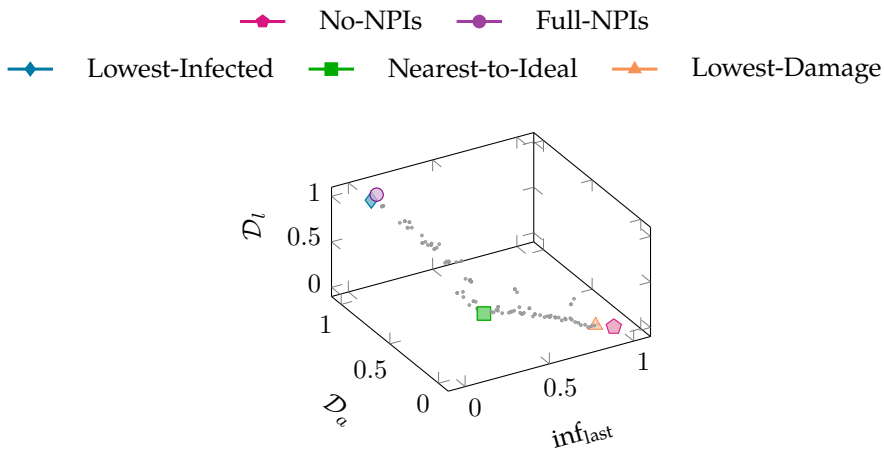
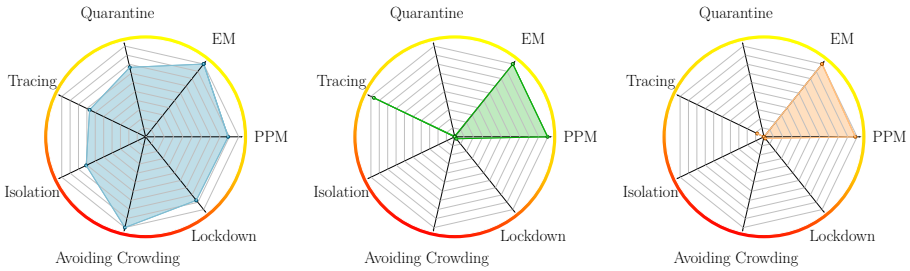


Figure 10.7: Pareto front evaluated by the algorithm NSGA-II for the BLE-Beacon data set using as objective function: $(\text{inf}_{\text{last}}, \mathcal{D}_a, \mathcal{D}_l)$.

sure ($\beta_{quarantine} = 0.01$) as we can consider the tracing protocol like an informed quarantine (see §Tracing in Section 10.2.3). It is interesting to observe that the configuration Nearest-to-Ideal can reduce by 50% the final number of infected with the only use of preventive measures. Clearly, more restrictive measures should be used in this kind of environment to reduce the contagion further. The application of this intervention brings to a social damage $\mathcal{D}_a$ of 0.36, meaning that each agent loses, on average, slightly more than a third of the individuals it would have met in normal conditions. The commuting damage $\mathcal{D}_l$ is 0.08, meaning that agents lose, on average, less than a tenth of the locations they would have normally visited.

As expected, the configuration achieving the lowest damage favors zero-damage interventions, like PPMs and EMs (see Figure 10.8c). Nevertheless, even though the final fraction of infected is lowered only by a small percentage, we can note that the overall trend (and the lowest peak) after introducing the intervention is generally reduced in comparison to the unmitigated scenario (see Figure 10.9). Regarding the configuration



(a) Lowest-Infected (b) Nearest-to-Ideal (c) Lowest-Damage

Figure 10.8: Combination of NPIs, corresponding to the configurations obtaining the lowest number of infected at the end of the simulation (Lowest-Infected), the lowest damage (Lowest-Damage), and the nearest to the optimal point (Nearest-to-Ideal) evaluated over the BLEBeacon data set.

obtaining the lowest fraction of infected, we can observe that this configuration selects the NPIs with the highest damage, located in the red zone of the radar chart (see Figure 10.8a). In particular, when this combination is applied, almost all rooms of the building become inaccessible ($\alpha_e = 0.87$), thus preventing the pathogen from spreading and bringing to very serious damage. For this reason, this configuration achieves the same results of the intervention implementing all protective measures (Full-NPIs).

The Foursquare Scenario

This paragraph discusses the outcomes for the Foursquare data set, resembling real-life movements of users of the Foursquare social network in Tokyo (see Section 10.4.4). Figures 10.10, 10.11, and 10.12 report the results.

As we can note from the Pareto front plotted in Figure 10.10, all solutions evaluated by the algorithm encode an NPI combination able to reduce the fraction of infected in the unmitigated scenario by at least 25%, paying highly variable damage. In this case, the Nearest-to-Ideal NPI

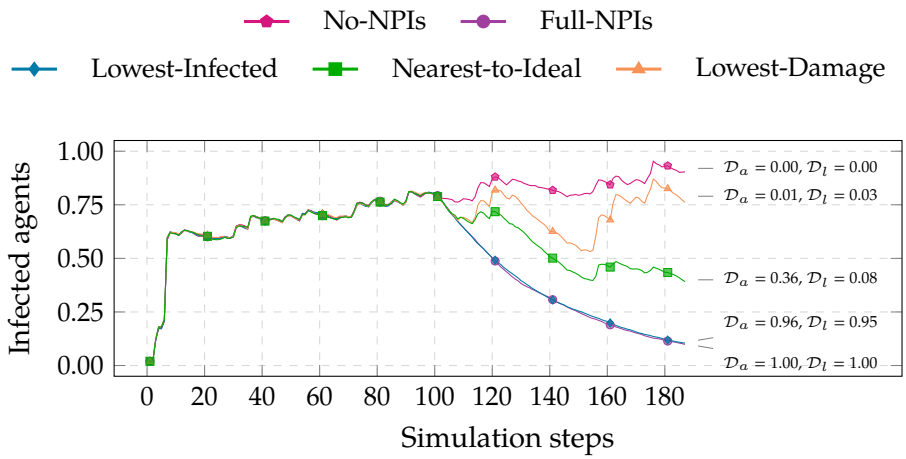


Figure 10.9: Epidemic trend in the BLEBeacon data set when (*i*) no NPIs (No-NPIs) and (*ii*) all NPIs (Full-NPIs) are implemented, (*iii*) the configuration bringing to the lowest number of infected at the end of the simulation is used (Lowest-Infected), (*iv*) the configuration obtaining the lowest damage is applied (Lowest-Damage), and (*v*) the configuration nearest to the optimum is used (Nearest-to-Ideal).

configuration (see Figure 10.11b) appears quite similar from the corresponding solution analyzed for the BLEbeacon data set. We can still observe a high value for $\alpha_p = 0.97$, meaning that the majority of the agents adopt PPMs and the use of sanitization measures. However, contrarily to BLEBeacon, α_i assumes the lower value of 0.013. Further, very small values are selected for the parameters $\beta_{isolation}$ and $\beta_{quarantine}$, equal to 0.002 and 0.06, respectively. Interestingly, this NPI combination can still halve the fraction of infected and hence decrease the epidemic trend (see Figure 10.12). At the same time, the damage brought by the intervention is minimal ($\mathcal{D}_a = 0.10$, $\mathcal{D}_l = 0.05$) and in practice due to the reduced percentage of quarantined and isolated agents. The most surprising aspect of this outcome is that despite the lack of lockdown measures, the epidemic trend still constantly diminishes. This result should be contextualized to the sparse nature of the data set; however, it suggests the fundamental role

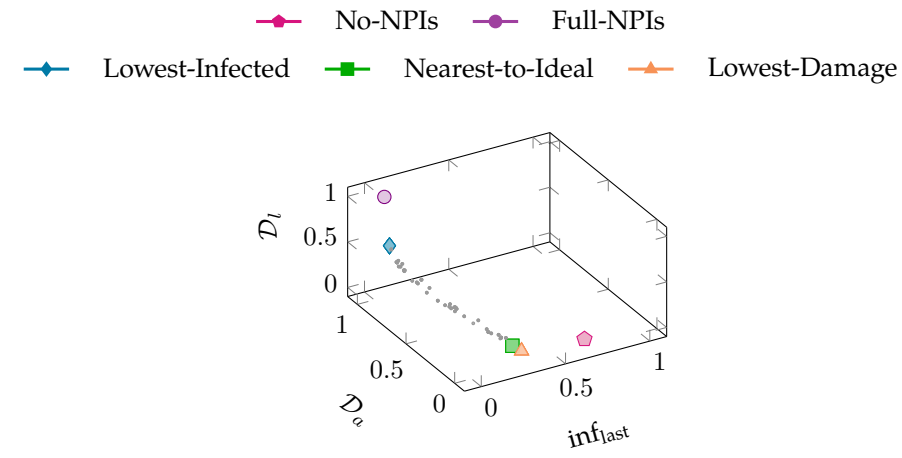
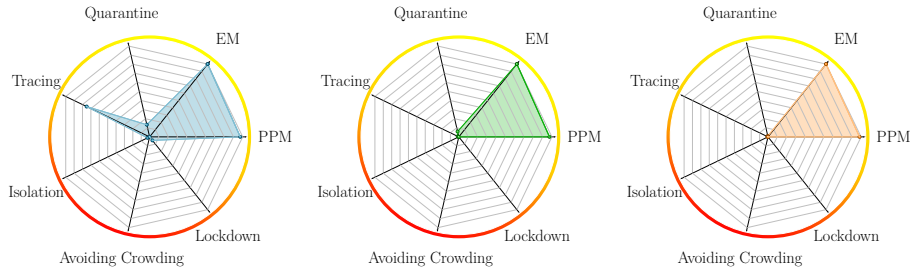


Figure 10.10: Pareto front evaluated by the algorithm NSGA-II for the Foursquare data set using the objective function: $(\inf_{\text{last}}, \mathcal{D}_a, \mathcal{D}_l)$.

of PPMs and EMs in reducing an epidemic spreading at zero-damage also in extremely crowded places.



(a) Lowest-Infected (b) Nearest-to-Ideal (c) Lowest-Damage

Figure 10.11: Combination of NPIs, corresponding to the configurations obtaining the lowest number of infected at the end of the simulation (Lowest-Infected, left), the lowest damage (Lowest-Damage, right), and the nearest to the optimal point (Nearest-to-Ideal, center) evaluated over the Foursquare data set.

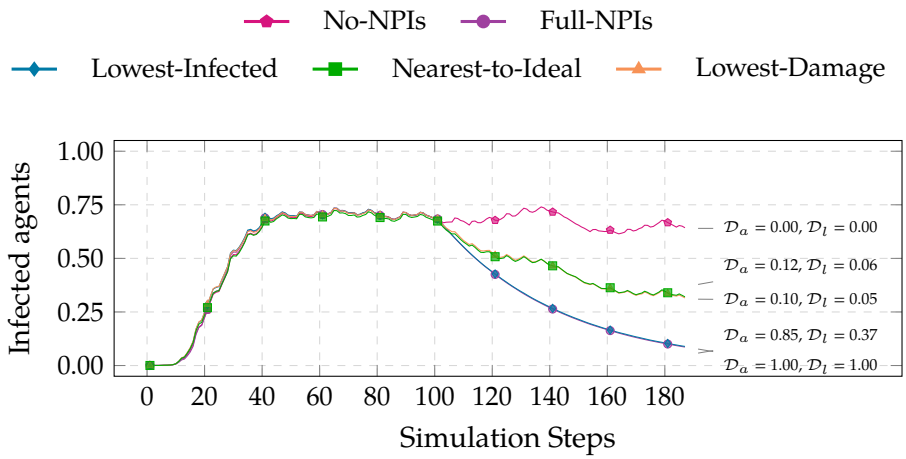


Figure 10.12: Epidemic trend in the Foursquare data set when i) no NPIs are implemented (No-NPIs), ii) all NPIs are in place (Full-NPIs), iii) the configuration bringing to the lowest number of infected at the end of the simulation is used (Lowest-Infected), iv) the configuration obtaining the lowest damage is applied (Lowest-Damage), and v) the configuration nearest to the optimum is used (Nearest-to-Ideal).

Regarding the configurations at the extremes of the Pareto front, the Lowest-Damage configuration almost corresponds to the Nearest-to-Ideal (see Figure 10.11c); while the Lowest-Infected combination (see Figure 10.11a) reaches the same effectiveness of Full-NPIs, with a lower commuting damage $\mathcal{D}_l$ and a non-significant difference for the social damage $\mathcal{D}_a$ (see Figure 10.12).

The GoT Scenario

This paragraph discusses the outcomes for the GoT data set, representing the characters' mobility pattern of the GoT TV series (see Section 10.4.4). Figures 10.13, 10.14, and 10.15 report the results.

As for the Foursquare data set, all the solutions elaborated by the genetic algorithm can at least halve the fraction of infected in the unmitigated scenario with variable damages (Figure 10.13). Focusing on the

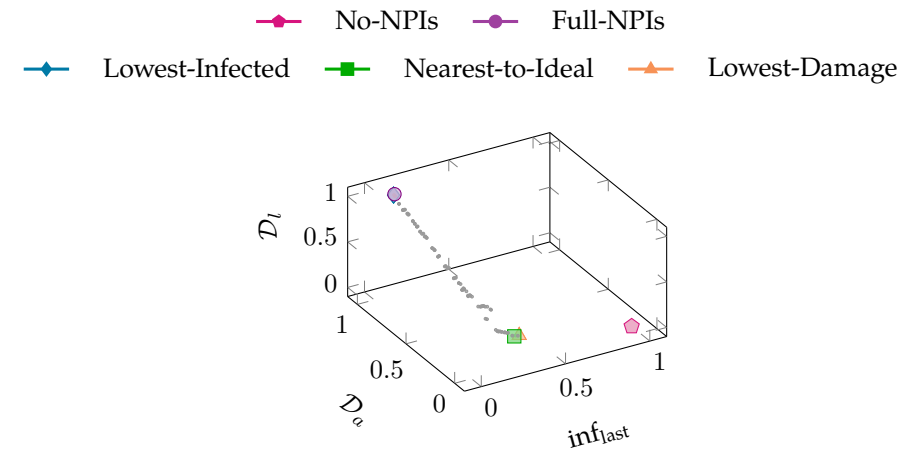
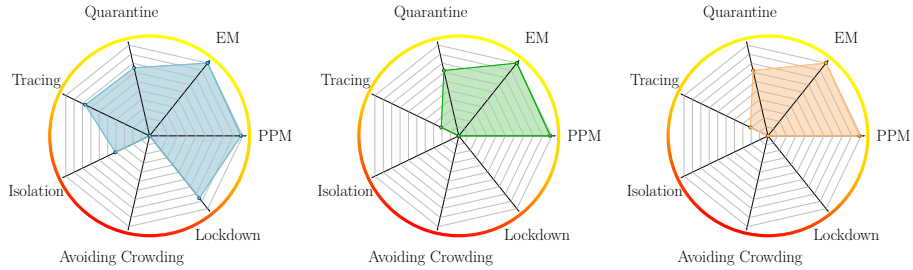


Figure 10.13: Pareto front evaluated by the algorithm NSGA-II for the GoT data set using the objective function: (inf_{last}, $\mathcal{D}_a$, $\mathcal{D}_l$).

configuration Nearest-to-Ideal (see Figure 10.14b), we can notice values similar to the solution evaluated for the BLEBeacon scenario. As before, almost all agents use PPMs ($\alpha_p = 0.98$) and locations are regularly sani-



(a) (Lowest-Infected (b) Nearest-to-Ideal (c) Lowest-Damage

Figure 10.14: Combination of NPIs, corresponding to the configurations obtaining the lowest number of infected at the end of the simulation (Lowest-Infected, left), the lowest damage (Lowest-Damage, right), and the nearest to the optimal point (Nearest-to-Ideal, center) evaluated over the GoT data set.

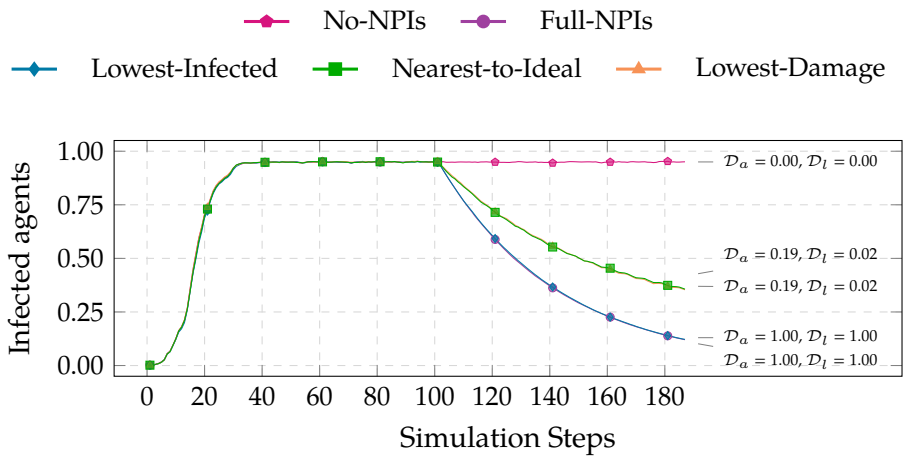


Figure 10.15: Epidemic trend in the GoT data set when i) no NPIs are implemented (No-NPIs), ii) all NPIs are in place (Full-NPIs), iii) the configuration bringing to the lowest number of infected at the end of the simulation is used (Lowest-Infected), iv) the configuration obtaining the lowest damage is applied (Lowest-Damage), and v) the configuration nearest to the optimum is used (Nearest-to-Ideal).

tized. On the other hand, we can note a lower use of tracing technologies, with only 21% of the agents adopting this measure, and a high probability of self-quarantine ($\beta_{quarantine} = 0.72$). Once again, the nature of the data set may explain the massive use of preventive measures, such as quarantine policies. In this scenario, only a limited number of characters move across most of the locations, thus coming in contact with many other characters. Hence, increasing the probability of entering the quarantine state corresponds to a higher chance to quarantine agents, especially the superspreader characters. Interestingly, this NPI combination can decrease by 65% the final fraction of infected without imposing strict restrictions that agents have to follow. Quantitatively speaking, we can observe a modest value for the social damage $\mathcal{D}_a$ (0.19) and a negligible commuting damage $\mathcal{D}_l$ (0.02).

As for the BLEBeacon scenario, the configuration Lowest-Infected per-

fectly overlaps with the Full-NPIs even though selecting slightly lower values for the NPI combination (see Figure 10.14a). Also in this case, the Lowest-Damage solution corresponds to the Nearest-to-Ideal (see Figure 10.14c).

10.6.3 Discussion

In the following, we sum up the most interesting elements from our experiments and discuss some real-world implications of these findings.

No NPIs can rule them all

If we focus our attention on the Nearest-to-Ideal NPI configuration, we can note how it largely varies across all data sets. For instance, we can observe that in the BLEBeacon data set, this NPI combination heavily counts on the probability of the agents to self-quarantine, translated into the massive use of tracing technology. We can observe a similar scenario in the GoT data set, in which the use of tracing measures is decreased, but the self-quarantine probability becomes considerable. Entirely opposite, the Nearest-to-Ideal configuration in the Foursquare scenario exploits a reduced quarantine probability and no tracing policies. Clearly, these results reflect the nature of the data fed into the simulation model. The BLEBeacon data set represents a closed environment; hence, it is easier for a pathogen to spread. As a consequence, stricter measures should be implemented. Although we shrank four months of check-ins into a single month timeframe, the Foursquare data set remains highly sparse, with few locations containing most user check-ins. As already observed, the sparse nature of this data may partially explain why the Nearest-to-Ideal solution selects little or zero values for isolation, quarantine, and tracing measures. Similarly, the mobility patterns encoded by the GoT data set make it reasonable to use tracing and quarantine measures, which are preventive interventions, as GoT characters tend to meet in communities, and only a few of them travel across many different places (see Section 10.4.4). In a real-world scenario, these results translate into ap-

plying different combinations of NPIs and tailoring their severity according to local specific data, like the current epidemiological situation and other socio-demographic features. This outcome is further aligned with the guidelines of the European Centre for Disease Prevention and Control [73] and with previous literature [188].

Please, wear your mask and wash your hands

Other than a widespread slogan to raise awareness in the population to fight the COVID-19, this phrase well describes another primary outcome of our experiments. Despite the different nature of the three data sets, we can easily spot that all the Nearest-to-Ideal NPI configurations share the application of PPMs and EMs, often selected in combination with other interventions (located in the yellow-orange zone of the radar chart, meaning that they generally cause low damage). Nonetheless, as discussed in the description of the results for the Foursquare experiment, the sole application of those interventions can still induce a decrease in the epidemic trend with the introduction of no damage. This result may have critical importance in a real-world situation as PPMs and EMs give a fundamental contribution in slowing the epidemic spreading at zero damage even in crowded locations and gathering places. However, the effectiveness of these policies remains strictly dependent upon the correct use made by the population.

Not a treasure hunt, but a guided search

As discussed in Section 5.4, GAs are widely exploited in ABMs to explore the best combination of parameters for the model. In the specific context of epidemic simulations, GAs may become a valuable tool to support policymakers as they can manually analyze some of the best solutions to control a pandemic and tune them according to specific needs. For instance, this approach can help investigate whether it is possible to implement an NPI combination that performs as well as closing gathering places. In this work, we examined GAs to optimize the implementation of NPIs while preserving two contrasting objectives: (i) the fraction of infected and (ii)

damage brought by the intervention. We considered two types of damage to include the need of people to meet other individuals ($\mathcal{D}_a$) and the possibility of visiting a given location ($\mathcal{D}_l$), e.g., simulating a person going to jog. GAs represents a general framework as the objective functions to optimize can be defined according to a precise need and based on the specific target to study. For example, an objective can model the economic damage caused by closing given business categories or the cost of hospitalizations. Similarly, our TVH framework can be tuned according to the specific compartmental model to implement, and the high-order network can be instantiated according to specific socio-demographic data.

Comment on the use of GAs

In this study, we used 100 individuals and 100 generations because of the long computational time required to run each simulation (e.g., forty minutes on average on the Foursquare data set) and because the implementation of the NSGA-II algorithm does not currently support the parallel evaluation of the individuals. However, the population size and the number of generations should be tuned to the complexity of the model when simulating real-world scenarios.

Simulation scalability

Scalability is a significant issue when dealing with massive agent-based simulations. For this reason, there is a strong line of research about designing and offering simulation engines able to run simulations in parallel/distributed environments (e.g., D-MASON [59]). In the specific case of epidemic simulations, the population size (in terms of number of agents) is usually limited, however the number of parameters may grow exponentially. In this sense, calibrating the model may become the most expensive computational part of the work. Having the scalable tools for model exploration is, thus,

critical.

10.7 Remarks

NPIs gained attention during the SARS-CoV-2 pandemic in 2019 as the only force to resist an unforeseen epidemic diffusion in the absence of effective pharmaceutical interventions. Further, despite the current availability of vaccines, a study performed on the UK region [137] stressed how the application of such measures cannot be completely relaxed as the current vaccination program alone is insufficient to contain the outbreak. Recent decisions of European Governments confirm this observation [33]. Thus, deeply understanding the potential impact of introducing regulation policies aiming to control and reduce the epidemic propagation is of fundamental importance to minimize the effect on both the economy and the psychological wellness of the society.

This chapter discussed how such controlling measures can be embedded within an epidemiological model based on high-order relationships between people and environments, mimicking direct and indirect contagion pathways over time. Specifically, we

- Provide a formal definition of NPIs, described by the WHO in [148], for our epidemiological framework based on TVHs (see Chapter 9);
- Evaluate each NPI applying the SIS compartmental model into an ABM that exploits our methodology (see Section 9.2.3) to simulate interactions between agents and environments by designing the agent mobility behavior according to real-world data sets;
- Design and implement a genetic algorithm-based methodology to optimize the choice of which NPI combination has to be adopted when contrasting objectives are considered.

The results of our experiments resemble previous literature and the guidelines of the WHO.

- Introduced alone, each NPI cannot extinguish an epidemic, even though some drastic measures, such as isolation and strict lockdown, have a higher impact in controlling the pathogen diffusion and considerably reduce its spread as indicated by the results in Section 10.5;
- The discussed outcomes further highlight that different combination of NPIs and their severity should be tailored according to local specific epidemiological data and that basic hygiene procedures are fundamental to reduce the spreading as presented in Section 10.6 and examined in Section 10.6.3.

Broadly translated, our findings indicate that the effects of NPIs in controlling are indisputable; however, the potential benefit of introducing NPIs in our society is functional to a massive and correct application in the population.

PART

V

Conclusions

Summary

11

Conclusions

Home is now behind you, the world is ahead!

The Hobbit
J. R. R. Tolkien

In short

In recent years, hypergraphs have continuously proved their ability to accurately abstract the high-order relations happening among the interactive parts of a real-world complex system. Their use in classical network science applications is still in its early development, and many research questions remain unaddressed. This dissertation fits in the broader context of (*i*) providing tools to directly and efficiently analyze hypergraphs and (*ii*) re-thinking diffusion phenomena by accounting for the high-order relations emerging from social interactions. Specifically, the contribution discussed in this dissertation embraces several research areas, from network analysis to agent-based modeling to the design and implementation of software frameworks. This final chapter summarizes the contributions, the achieved research outcomes and delineates the potential impact of this research. It finally discusses possible future directions on each specific topic addressed.

11.1 Remarks

The following sections detail the specific problems addressed, the models and solutions proposed, and the main outcomes obtained. All software produced is freely available on GitHub public repositories to improve the reproducibility of existing work and allow a fair comparison with other methods.

Tools for hypergraphs

Motivated by the lack of a comprehensive and efficient hypergraph-specific library and the need for software libraries designed to perform operations directly on hypergraphs, we developed SimpleHypergraphs.jl, a software library to model, analyze, and visualize hypergraphs, written in Julia and designed for high-performance computing. In this dissertation, we described the main motivations behind creating SimpleHypergraphs.jl, the library's design choices, and its memory model. We further illustrated the functionalities offered by the software, including graph transformations and hypergraph visualization methods. We also presented two case studies with the twofold objective of demonstrating how it is possible to exploit the proposed library and comparing hypergraphs with their corresponding graph counterpart to explore whether high-order structures convey more information in addressing specific tasks. Contextually, we also described a generalized version of the label propagation algorithm for community detection suitable for hypergraphs.

The development of SimpleHypergraphs.jl is a joint project with the Warsaw School of Economics (Warsaw, Poland) and the Ryerson University (Toronto, Canada). This library supports the adoption of hypergraphs in the network science community by enabling ready-to-use methods to model, manipulate, analyze, and visualize such high-order structures. Today, SimpleHypergraphs.jl is the reference Julia library for working with hypergraphs.

Social Influence on High-order Networks

Social influence involves intentional and unintentional efforts to change another person's beliefs, attitudes, or behavior. With the advent of online social networks, such a phenomenon assumed even more importance given the exponential number of people a user can reach through them. Traditionally, algorithmic questions in information diffusion research have been formalized on graphs. However, friends and, more in general, communities exert influence on individuals, which can modify their behavior accordingly. Hypergraphs can, hence, be used to account for these high-order interactions.

Based on this consideration, we proposed a new linear threshold high-order diffusion model that mimics real-world social dynamics, where individuals influence the group they belong to, but - in turn - the group itself influences their choices. We further introduced the formal definition of the Target Set Selection problem on hypergraphs (TSSH), a key algorithmic question in information diffusion research, whose goal is to find the smaller set of vertices that can influence the whole network according to the diffusion model defined. Since the TSSH problem is NP-hard, we described four heuristics to address it and extensively evaluated these algorithms on random and real-world networks.

Loosely speaking, the results obtained suggest that the choice of which procedure should be used strictly depends on the characteristics of the hyper-networks, such as the density of the network or the distribution of both vertex degrees and hyperedge sizes.

Research about social diffusion phenomena has practical implications and impacts both society and industry. For instance, online social networks represent a critical application domain as, today, they embody one of the most effective media to share information. Nevertheless, when it comes to fake news, it is of fundamental importance to block its diffusion. Further, studying diffusion processes also directly applies to detecting negative influence towards vicious and harmful behavior, which is not so difficult to find in social or games communities. The existence of

several patents[1] from big tech companies related to the spreading phenomena confirms the importance of investigating such topics also for the industry landscape. For example, finding key users to promote a product within social media - to make sure that more and more people start using it - is a typical example but still critical in companies' marketing strategies.

Epidemic Dynamics on Temporal High-order Networks

Influence diffusion models are usually treated as specific cases of epidemic models. Nevertheless, in this context, using models as realistic as possible to foresee and possibly control a pathogen's diffusion is even more important. Towards this, hypergraphs represent a valid tool to account for community structure and infection pressure.

From the perspective of epidemic dynamics, this dissertation motivated the use of (temporal) hypergraphs rather than (temporal) graphs to analyze epidemic spreading processes. We then introduced the formal definition of temporal hypergraphs, described a high-order SIS compartmental equation model suitable for TVHs, and discussed how we assembled these elements into an agent-based framework. We further presented a sensitivity analysis of the TVH model to the epidemic parameters and different discretization of the time intervals when direct or indirect contacts may happen.

Results suggest two main outcomes. First, exploiting TVHs may improve the accuracy of the estimation of an epidemic diffusion; second, correctly modeling the time interval within which direct or indirect contagion may happen is critical to not overestimate either contagion type.

Built on top of the TVH model, we also proposed a fine-grain modeling methodology for Non-Pharmaceutical Interventions (NPIs). We then delved into reviewing personal protective, environmental, and social distancing measures and how they can be embedded into an epidemiological model based on high-order networks, ABMs, and the SIS equation-based model. We further described how we formally enriched the TVH mod-

[1] `https://www.wipo.int`

eling framework to support the evaluation of NPIs. After assessing the ability of each intervention in controlling an epidemic propagation, we discussed a multi-objective optimization framework, which, based on the epidemiological data, calculates the NPI combination that should be implemented to minimize the spread of an epidemic as well as the damage due to the intervention.

Broadly translated, our findings indicate that the effects of NPIs in controlling an epidemic are indisputable; however, the potential benefit of their introduction in our society is functional to a massive and correct application in the population.

The impact of research about epidemic-spreading phenomena became evident since the beginning of the current COVID-19 pandemic. Working towards more realistic epidemic diffusion models could allow us to discover key spreaders and immunize them effectively. These choices could, in turn, inform policy-making systems and processes to avoid critical scenarios and the application of the strictest NPIs.

11.2 Future Work

The topics discussed in this dissertation leave room for many interesting future developments. These lines of inquiry span different research directions and aim to exploit different solution paradigms, for instance, exploring learning techniques, like deep learning (DL) [85] for networks and reinforcement learning (RL) [174].

Coding hypergraphs. SimpleHypergraphs.jl is under active development. In 2019, the library was presented to the JuliaCon conference[2] and currently stands as the standard library to manipulate hypergraphs in Julia. We plan to continue integrating hypergraph-specific algorithms in SimpleHypergraphs.jl to provide a comprehensive framework comparable to the state-of-the-art graph libraries. We further plan to realize a comprehensive and interactive hypergraph reposi-

[2] `https://www.youtube.com/watch?v=8Zfv9bySFBw`

tory to provide the community with a tool to easily access domain-specific hypergraphs and their basic structural properties (e.g., along the lines of Network Repository[3] and SNAP[4]).

Influence diffusion on hypergraphs. Looking at the social influence research field, first, the same procedures discussed in this dissertation may be exploited to measure influence diffusion over groups (hyperedges) rather than over single individuals (vertices). In real-world or online social networks, groups play an essential role in understanding people's personal behaviors. Therefore, it is critical to systematically study how the many influences the single person when final decisions depend upon groups of people. Second, motivated by the lack of extensive studies on information diffusion processes on high-order structures, another research path could be developing other diffusion models suitable for hypergraphs. Having several approaches to analyze influence diffusion patterns will allow using the most suitable model according to the interactions happening in the real world. Finally, another interesting line of inquiry would be studying diffusion phenomena in heterogeneous hypergraphs, where both vertices and relations may be of more than one type, thus giving birth to different diffusion processes that may influence each other.

Epidemic dynamics via hypergraphs. Regarding the study of epidemic dynamics via hypergraphs, a crucial research line is the design, development, and analysis of immunization strategies that exploit the high-order nature of the underlying contact network to more efficiently and effectively identify and block super-spreaders. This problem has been traditionally tackled in network science using heuristics based on some structural properties of the contact network [186]. Other than testing the effectiveness of these heuristics on hypergraphs, we could also explore DL techniques and RL approaches

[3] `http://networkrepository.com/`
[4] `http://snap.stanford.edu/data`

to find the most critical vertices. Clearly, this problem and its possible solution methodologies straightforwardly transfer to the social influence diffusion domain. A second helpful contribution is the development of other compartmental models to include in the TVH framework, within which higher-order diffusion phenomena can be examined. As for the study of influence phenomena, having different diffusion models will enable the use of the most suitable real-world abstraction.

Hypergraphs have risen to prominence in the academic and industry landscape in the last ten years. The new computational possibilities and the consciousness that high-order interactions cannot be overlooked when modeling and studying real-world systems have given a start to new research areas and renewed efforts in topics where graphs dominated. Many-to-many relationships are everywhere in the world around us, and the need to *re-think high-order* embodies a paradigm shift that will reshape and expand network science as we know it.

Bibliography

[1] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44):4017 – 4022, 2010. [Cited on page 52.]

[2] A. Agarwal, A. Corvalan, J. Jensen, and O. Rambow. Social Network Analysis of Alice in Wonderland. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 88–96, Montréal, Canada, 2012. Association for Computational Linguistics. [Cited on page 105.]

[3] N. Ahmed, R.A. Michelin, W. Xue, S. Ruj, R. Malaney, S.S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and S.K. Jha. A Survey of COVID-19 Contact Tracing Apps. *IEEE Access*, 8:134577–134601, 2020. [Cited on page 204.]

[4] S. G. Aksoy, C. Joslyn, C. Ortiz Marrero, B. Praggastis, and E. Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1):16, 2020. [Cited on pages 5, 17, 24, 25, 26, 29, 40, and 116.]

[5] R. Alberich, J. Miro-Julia, and F. Rossello. Marvel Universe looks almost like a real social network, 2002. [Cited on page 106.]

[6] J. E. Aledort, N. Lurie, J. Wasserman, and S. A. Bozzette. Non-pharmaceutical public health interventions for pandemic influenza: an evaluation of the evidence base. *BMC Public Health*, 7(1):208, 2007. [Cited on page 67.]

[7] A. Aleta, D. Martín-Corral, M. A. Bakker, A. Pastore y Piontti, M. Ajelli, M. Litvinova, M. Chinazzi, N. E. Dean, M. E. Halloran, I. M. Longini, A. Pentland, A. Vespignani, Y. Moreno, and E. Moro. Quantifying the importance and location of sars-cov-2 transmission events in large metropolitan areas. *medRxiv*, 2020. [Cited on pages 68, 179, and 232.]

[8] A. Aleta, D. Martín-Corral, A. Pastore y Piontti, M. Ajelli, M. Litvinova, M. Chinazzi, N. E. Dean, M. E. Halloran, I. M. Longini Jr, S. Merler, A. Pentland, A. Vespignani, E. Moro, and Y. Moreno. Modelling the impact of testing, contact tracing and household quarantine on second waves of covid-19. *Nature Human Behaviour*, 2020. [Cited on pages 68, 179, and 232.]

[9] M.J. Alves and M. Almeida. MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 34(11):3458–3470, 2007. [Cited on page 71.]

[10] P. W. Anderson. More Is Different. *Science*, 177, 1972. [Cited on page 3.]

[11] A. Antelmi, J. Breslin, and K. Young. Understanding User Engagement with Entertainment Media: A Case Study of the Twitter Behaviour of Game of Thrones (GoT) Fans. In *2018 IEEE Games, Entertainment, Media Conference*, GEM'18, pages 1–9, 2018. [Cited on page 107.]

[12] A. Antelmi, G. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, and P. Szufel. SimpleHypergraphs.jl - a Julia Software Framework for Modeling and Analyzing Hypergraphs. `https://https://github.com/pszufe/SimpleHypergraphs.jl`, 2019. [Online; 2021]. [Cited on pages 82 and 84.]

[13] A. Antelmi, G. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, and P. Szufel. SimpleHypergraphs.jl—Novel Software Framework for Modelling and Analysis of Hypergraphs. In *Algorithms and Models for the Web Graph, WAW'19*, WAW'19, pages 115–129, Cham, 2019. Springer International Publishing. [Cited on page 11.]

[14] A. Antelmi, G. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, and P. Szufel. Analyzing, Exploring, and Visualizing Complex Networks via Hypergraphs using SimpleHypergraphs.jl. *Internet Mathematics*, 2020. [Cited on pages 11 and 27.]

[15] A. Antelmi, G. Cordasco, V. Scarano, and C. Spagnuolo. Modeling and Evaluating Epidemic Control Strategies With High-Order Temporal Networks. *IEEE Access*, 9:140938–140964, 2021. [Cited on page 11.]

[16] A. Antelmi, G. Cordasco, C. Spagnuolo, and V. Scarano. A Design-Methodology for Epidemic Dynamics via Time-Varying Hypergraphs. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 61–69. International Foundation for Autonomous Agents and Multiagent Systems, 2020. [Cited on page 11.]

[17] A. Antelmi, G. Cordasco, C. Spagnuolo, and P. Szufel. Information Diffusion in Complex Networks: A Model Based on Hypergraphs and Its Analysis. In *Algorithms and Models for the Web Graph, WAW'20*, Lecture Notes in Computer Science, pages 36–51, Cham, 2020. Springer International Publishing. [Cited on page 11.]

[18] A. Antelmi, G. Cordasco, C. Spagnuolo, and P. Szufel. Social Influence Maximization in Hypergraphs. *Entropy*, 23(7), 2021. [Cited on page 11.]

[19] A. Antelmi, D. Malandrino, and V. Scarano. Characterizing the Behavioral Evolution of Twitter Users and The Truth Behind the 90-9-1 Rule. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 1035–1038, New York, NY, USA, 2019. Association for Computing Machinery. [Cited on pages 190 and 213.]

[20] Antelmi, A., Cordasco, G., Spagnuolo, C. and Szufel, P. Social Influence Maximization in Hypergraphs - Data set, 2021. [Cited on page 145.]

[21] Antelmi, A., Cordasco, G., Spagnuolo, C. and Szufel, P. Social Influence Maximization in Hypergraphs - Julia Package, 2021. [Cited on page 142.]

[22] H. Arora, T. S. Raghu, and A. Vinze. Decision Support for Containing Pandemic Propagation. *ACM Transactions on Management Information Systems*, 2(4), 2012. [Cited on pages 60, 61, and 68.]

[23] Askwith, I.D. Television 2.0 : reconceptualizing TV as an engagement medium. `dspace.mit.edu/handle/1721.1/41243`, 2007. [Online; 2019]. [Cited on page 107.]

[24] A. Atalan. Is the lockdown important to prevent the COVID-19 pandemic? Effects on psychology, environment and economy-perspective. *Annals of Medicine and Surgery*, 56:38 – 42, 2020. [Cited on page 199.]

[25] C. Avin, Z. Lotker, Y. Nahum, and D. Peleg. Random Preferential Attachment Hypergraph. In *2019 IEEE/ACM International Conference on Advances*

in Social Networks Analysis and Mining, ASONAM'19, pages 398–405, 2019. [Cited on page 87.]

[26] S. Banerjee, M. Jenamani, and D. K. Pratihar. A survey on influence maximization in a social network. *Knowledge and Information Systems*, 62(9):3417–3455, 2020. [Cited on pages 50, 51, and 52.]

[27] S. Bansal, B. T. Grenfell, and L. A. Meyers. When individual behaviour matters: homogeneous and network models in epidemiology. *Journal of The Royal Society Interface*, 4(16):879–891, 2007. [Cited on page 172.]

[28] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. [Cited on page 4.]

[29] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11):3747–3752, 2004. [Cited on page 4.]

[30] A. Barrat, M. Barthélemy, and A. Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, 2008. [Cited on page 8.]

[31] M. Barthélemy, A. Barrat, R. Pastor-Satorras, and A. Vespignani. Characterization and modeling of weighted networks. *Physica A: Statistical Mechanics and its Applications*, 346(1):34–43, 2005. [Cited on page 4.]

[32] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G Young, and G. Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020. Networks beyond pairwise interactions: Structure and dynamics. [Cited on pages 5, 6, 8, 38, 57, 62, and 64.]

[33] BBC. Austria to go into full lockdown as Covid surges. [Cited on page 246.]

[34] A. Bellaachia and M. Al-Dhelaan. Random Walks in Hypergraph. *International Journal of Education and Information Technologies (NAUN)*, 15:13–20, 2021. [Cited on page 93.]

[35] A.R. Benson. Austin R. Benson Research Data Sets. `https://www.cs.cornell.edu/~arb/data/`. [Cited on pages 73, 74, 75, 77, 144, and 145.]

[36] C. Berge. *Hypergraphs: combinatorics of finite sets*, volume 45 of *North-Holland Mathematical Library*. North-Holland, 1984. [Cited on page 16.]

[37] A. Beveridge. Network of Thrones. `networkofthrones.wordpress.com`, 2019. [Online; 2019]. [Cited on pages 106 and 113.]

[38] A. Beveridge and J. Shan. Network of Thrones. *Math Horizons*, 23(4):18–22, 2016. [Cited on pages 105 and 107.]

[39] J. Bezanson, S. Karpinski, V. Shah, and A. Edelman. The Julia Programming Language. `https://julialang.org/`, 2012. [Online; 2021]. [Cited on page 43.]

[40] S. Boccaletti, G. Bianconi, R. Criado, C.I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014. The structure and dynamics of multilayer networks. [Cited on page 4.]

[41] Á. Bodó, G. Y. Katona, and Pé. L. Simon. SIS Epidemic Propagation on Hypergraphs. *Bulletin of Mathematical Biology*, 78:713–735, 2016. [Cited on pages 63, 173, 181, 184, and 191.]

[42] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, 2002. [Cited on page 199.]

[43] A. Bonato, D.R. D'Angelo, E.R. Elenberg, D.F. Gleich, and Y. Hou. Mining and Modeling Character Networks. In *Algorithms and Models for the Web Graph*, WAW'16, pages 100–114, Cham, 2016. Springer International Publishing. [Cited on page 105.]

[44] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, page 946–957, USA, 2014. Society for Industrial and Applied Mathematics. [Cited on pages 53 and 141.]

[45] A. Bouchnita and A. Jebrane. A hybrid multi-scale model of COVID-19 transmission dynamics to assess the potential of non-pharmaceutical interventions. *Chaos, Solitons & Fractals*, 138:109941, 2020. [Cited on pages 69, 207, and 232.]

[46] F. Brauer. Mathematical epidemiology: Past, present, and future. *Infectious Disease Modelling*, 2(2):113–127, 2017. [Cited on page 61.]

[47] A Bretto. *Hypergraph Theory: An Introduction*. Mathematical Engineering. Springer International Publishing, New York, NY (USA), 2013. [Cited on pages 15, 17, 28, and 175.]

[48] D. Butcher. Julia programming language: this is the new skill hedge funds are asking for. `https://www.efinancialcareers.co.uk/news/2016/06/boost-your-salary-by-learning-the-julia-programming-language`, 2016. [Online; 2021]. [Cited on page 43.]

[49] C.T. Butts. Revisiting the Foundations of Network Analysis. *Science*, 325(5939):414–416, 2009. [Cited on page 4.]

[50] M. Carillo, G. Cordasco, F. Serrapica, V. Scarano, C. Spagnuolo, and P. Szufel. Distributed simulation optimization and parameter exploration framework for the cloud. *Simulation Modelling Practice and Theory*, 83:108–123, 2018. [Cited on page 70.]

[51] G. Casati. iper. `github.com/fibo/iper`, 2017. [Online; 2021]. [Cited on page 41.]

[52] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-Varying Graphs and Dynamic Networks. In *Ad-hoc, Mobile, and Wireless Networks*, pages 346–359, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. [Cited on page 174.]

[53] N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009. [Cited on page 52.]

[54] W. Chen, L.V.S. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks*. Morgan & Claypool Publishers, 2013. [Cited on page 133.]

[55] U. Chitra and B. Raphael. Random Walks on Hypergraphs with Edge-Dependent Vertex Weights. In *Proceedings of the International Conference on Machine Learning*, volume 97 of *ICML'19*, pages 1172–1181, 2019. [Cited on page 22.]

[56] M. L. Chouder, S. Rizzi, and R. Chalal. JSON Datasets for Exploratory OLAP. In *Mendeley Data*, volume V1, 2017. [Cited on pages 75, 77, 144, and 145.]

[57] G. Cordasco, R. De Chiara, F. Raia, V. Scarano, C. Spagnuolo, and L. Vicidomini. Designing computational steering facilities for distributed agent based simulations. In *SIGSIM Principles of Advanced Discrete Simulation*, SIGSIM-PADS'13, pages 385–390, 2013. [Cited on page 62.]

[58] G. Cordasco, L. Gargano, M. Mecchia, A.A. Rescigno, and U. Vaccaro. Discovering Small Target Sets in Social Networks: A Fast and Effective Algorithm. *Algorithmica*, 80(6):1804–1833, 2018. [Cited on pages 52, 137, and 139.]

[59] G. Cordasco, C. Spagnuolo, and V. Scarano. Toward the New Version of D-MASON: Efficiency, Effectiveness and Correctness in Parallel and Distributed Agent-Based Simulations. In *IEEE International Parallel and Distributed Processing Symposium Workshops*, 2016. [Cited on page 245.]

[60] Cuebiq. Cuebiq, 2022. [Cited on page 178.]

[61] G. F. de Arruda, G. Petri, and Y. Moreno. Social contagion models on hypergraphs. *Physical Review Research*, 2:023032, 2020. [Cited on pages 66 and 185.]

[62] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. [Cited on pages 71 and 232.]

[63] Ministero della Salute. Covid-19, travellers, 2020. [Cited on page 205.]

[64] F. Dignum, V. Dignum, P. Davidsson, A. Ghorbani, M. van der Hurk, M. Jensen, C. Kammler, F. Lorig, L.G. Ludescher, A. Melchior, R. Mellema, C. Pastrav, L. Vanhee, and H. Verhagen. Analysing the Combined Health, Social and Economic Impacts of the Corovanvirus Pandemic Using Agent-Based Social Simulation. *Minds and Machines*, 30(2):177–194, 2020. [Cited on page 69.]

[65] P. Domingos and M. Richardson. Mining the Network Value of Customers. In *Proceedings of the Seventh ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, KDD'01, page 57–66, New York, NY, USA, 2001. Association for Computing Machinery. [Cited on page 51.]

[66] D. Duperron. hypergraph. `https://github.com/yamafaktory/hypergraph`, 2021. [Online; 2021]. [Cited on page 40.]

[67] K.T.D. Eames, C.. Webb, K. Thomas, J. Smith, R. Salmon, and J.M.F. Temple. Assessing the role of contact tracing in a suspected H7N2 influenza A outbreak in humans in Wales. *BMC infectious diseases*, 10:141–141, 2010. [Cited on page 204.]

[68] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge Books. Cambridge University Press, Cambridge (UK), 2010. [Cited on pages 61 and 174.]

[69] M. Edelson, T. Sharot, R. J. Dolan, and Y. Dudai. Following the crowd: brain substrates of long-term memory conformity. *Science*, 333(6038):108–111, 2011. [Cited on page 128.]

[70] M. Ehrhardt, J. Gašper, and S. Kilianová. SIR-based mathematical modeling of infectious diseases with vaccination and waning immunity. *Journal of Computational Science*, 37:101027, 2019. [Cited on page 172.]

[71] H. Fang, L. Wang, and Y. Yang. Human mobility restrictions and the spread of the Novel Coronavirus (2019-nCoV) in China. *Journal of Public Economics*, 191:104272, 2020. [Cited on page 199.]

[72] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491), 2020. [Cited on pages 202 and 204.]

[73] European Centre for Disease Prevention and Control. Guidelines for non-pharmaceutical interventions to reduce the impact of COVID-19 in the EU/EEA and the UK. Ecdc technical report, European Centre for Disease Prevention and Control, 2020. [Cited on page 244.]

[74] European Centre for Disease Prevention and Control (ECDC). Guide to public health measures to reduce the impact of influenza pandemics in Europe: 'The ECDC menu'., 2009. [Cited on page 203.]

[75] D.R. Forsyth. *Group Dynamics*. Cengage Learning, Boston, MA (USA), 2019. [Cited on page 128.]

[76] M.C. Fu, F.W. Glover, and J. April. Simulation optimization: a review, new developments, and applications. In *Proceedings of the Winter Simulation Conference.*, WSC'05, page 13, 2005. [Cited on pages 70 and 71.]

[77] S. Funk, M. Salathé, and V. A. A. Jansen. Modelling the influence of human behaviour on the spread of infectious diseases: a review. *Journal of The Royal Society Interface*, 7(50):1247–1256, 2010. [Cited on page 67.]

[78] V. Gangal, B. Ravindran, and R. Narayanam. HEMI: hyperedge majority influence maximization. *Proceedings of the 2nd International Workshop on Social Influence Analysis*, abs/1606.05065, 2016. [Cited on pages 54, 140, and 141.]

[79] N. Ganguly, T. Krueger, A. Mukherjee, and S. Saha. Epidemic spreading through direct and indirect interactions. *Physical Review E*, 90:032808, 2014. [Cited on page 172.]

[80] R. H. Gass. Sociology of Social Influence. In James D. Wright, editor, *International Encyclopedia of the Social & Behavioral Sciences*, pages 348–354. Elsevier, Oxford, second edition, 2015. [Cited on page 128.]

[81] G. F. Gensini, M. H. Yacoub, and A. A. Conti. The concept of quarantine in history: from plague to SARS. *Journal of Infection*, 49(4):257 – 261, 2004. [Cited on page 198.]

[82] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri. Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nature Medicine*, 26(6):855–860, 2020. [Cited on page 209.]

[83] J. Goldenberg, B. Libai, and E. Muller. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters*, 12(3):211–223, 2001. [Cited on page 51.]

[84] W. Gong, E.-P. Lim, and F. Zhu. Characterizing Silent Users in Social Media Communities. In *Proceedings of the Ninth International Conference on Web and Social Media*, ICWSM'15, pages 140–149. AAAI Press, 2015. [Cited on pages 190 and 213.]

[85] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. [Cited on page 255.]

[86] Italian Government. Immuni, 2020. [Cited on page 209.]

[87] L. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978. [Cited on page 50.]

[88] M. R. Gualano, G. Lo Moro, G. Voglino, F. Bert, and R. Siliquini. Effects of Covid-19 Lockdown on Mental Health and Sleep Disturbances in Italy. *International Journal of Environmental Research and Public Health*, 17(13):4779, 2020. [Cited on page 199.]

[89] R. A. Hammond. Considerations and Best Practices in Agent-Based Modeling to Inform Policy. In R. Wallace, A. Geller, and V.A. Ogawa, editors, *Committee on the Assessment of Agent-Based Models to Inform Tobacco Product Regulation. Board on Population Health and Public Health Practice; Institute of Medicine*, pages Appendix A, 161–193. National Academic Press (US), Washington DC, WA, US, 2015. [Cited on page 172.]

[90] J. Harant, A. Pruchnewski, and M. Voigt. On dominating sets and independent sets of graphs. *Combinatorics, Probability and Computing*, 8(6):547–553, 1999. [Cited on page 52.]

[91] HBO. Game of Thrones. `www.hbo.com/game-of-thrones`, 2019. [Online; 2019]. [Cited on page 107.]

[92] D. J. Higham and H.-L. de Kergorlay. Epidemics on hypergraphs: spectral thresholds for extinction. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2252):20210232, 2021. [Cited on pages 66 and 186.]

[93] N. Hoertel, M. Blachier, C. Blanco, M. Olfson, M. Massetti, M. Sánchez Rico, F. Limosin, and H. Leleu. A stochastic agent-based model of the SARS-CoV-2 epidemic in France. *Nature Medicine*, 26(9):1417–1421, 2020. [Cited on pages 68, 179, and 232.]

[94] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012. Temporal Networks. [Cited on page 4.]

[95] K. Huang, S. Wang, G. Bevilacqua, X. Xiao, and L.V.S. Lakshmanan. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *Proceedins of VLDB Endowment*, 10(9):913–924, 2017. [Cited on page 52.]

[96] X. Huang, D. Chen, T. Ren, and D. Wang. A survey of community detection methods in multilayer networks. *Data Mining and Knowledge Discovery*, 35(1):1–45, 2021. [Cited on page 91.]

[97] B. Hunter, E.and Mac Namee and J. D. Kelleher. A Taxonomy for Agent-Based Models in Human Infectious Disease Epidemiology. *Journal of Artificial Societies and Social Simulation*, 20(3):2, 2017. [Cited on page 172.]

[98] I. Iacopini, G. Petri, A. Barrat, and V. Latora. Simplicial models of social contagion. *Nature Communications*, 10(1):2485, 2019. [Cited on pages 65 and 66.]

[99] H. Inoue and Y. Todo. The propagation of economic impacts through supply chains: The case of a mega-city lockdown to prevent the spread of COVID-19. *PLOS ONE*, 15(9):e0239251, 2020. [Cited on page 69.]

[100] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3):392–403, 1998. [Cited on page 71.]

[101] A. Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002. [Cited on page 71.]

[102] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, and A. Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018. [Cited on page 103.]

[103] Jeffrey, L. Game of Thrones Datasets and Visualizations. `https://github.com/jeffreylancaster/game-of-thrones`, 2019. [Cited on pages 76, 107, 144, 145, 211, and 214.]

[104] L. Jenkins, T. Bhuiyan, S. Harun, C. Lightsey, D. Mentgen, S. Aksoy, T. Stavcnger, M. Zalewski, H. Medal, and C. Joslyn. Chapel HyperGraph Library (CHGL). In *2018 IEEE High Performance extreme Computing Conference*, HPEC'18, pages 1–6. IEEE, 2018. [Cited on page 38.]

[105] L. Jenkins, T. Bhuiyan, S. Harun, C. Lightsey, D. Mentgen, S. Aksoy, T. Stavcnger, M. Zalewski, H. Medal, and C. Joslyn. Chapel Hypergraph Library (CHGL). `github.com/pnnl/chgl`, 2018. [Online; 2021]. [Cited on pages 38 and 84.]

[106] B. Jhun, M. Jo, and B. Kahng. Simplicial SIS model in scale-free uniform hypergraph. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):123207, 2019. [Cited on pages 65 and 185.]

[107] W. Jiang, J. Qi, J. X. Yu, J. Huang, and R. Zhang. HyperX. `github.com/jinhuang/hyperx`, 2015. [Online; 2021]. [Cited on pages 41 and 116.]

[108] W. Jiang, J. Qi, J. X. Yu, J. Huang, and R. Zhang. HyperX: A Scalable Hypergraph Framework. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):909–922, 2019. [Cited on page 41.]

[109] A. Jindal and S. Rao. Agent-Based Modeling and Simulation of Mosquito-Borne Disease Transmission. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 426–435, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems. [Cited on pages 62 and 172.]

[110] C.A. Joslyn, S. Aksoy, D. Arendt, L. Jenkins, B. Praggastis, E. Purvine, and M. Zalewski. Hypergraph Analytics of Domain Name System Relationships. In *Proceedings of Algorithms and Models for the Web Graph - 17th International Workshop (WAW'20)*, volume 12091 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2020. [Cited on page 40.]

[111] JuliaLang.org. JuliaLang Repository. `https://github.com/JuliaLang/julia`, 2012. [Online; 2021]. [Cited on page 47.]

[112] JuliaLang.org. Julia Micro-Benchmarks. `https://julialang.org/benchmarks/`, 2021. [Online; 2021]. [Cited on page 44.]

[113] B. Kamiński, V. Poulin, P. Prałat, P. Szufel, and F. Théberge. Clustering via hypergraph modularity. *PloS one*, 14(11), 2019. [Cited on pages 91 and 102.]

[114] V. Karve. multihypergraph. `github.com/vaibhavkarve/multihypergraph`, 2018. [Online; 2021]. [Cited on page 42.]

[115] S. Katoch, S.S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2020. [Cited on page 71.]

[116] G.O.H. Katona. Extremal Problems for Hypergraphs. In *Combinatorics*, pages 215–244, Dordrecht, 1975. Springer Netherlands. [Cited on page 17.]

[117] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'03, page 137–146, New York, NY, USA, 2003. Association for Computing Machinery. [Cited on pages 51 and 129.]

[118] M. Kim, D. Newth, and P. Christen. Modeling Dynamics of Diffusion Across Heterogeneous Social Networks: News Diffusion in Social Media. *Entropy*, 15(10):4215–4242, 2013. [Cited on page 52.]

[119] 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, 2014. [Cited on page 4.]

[120] D. Klinkenberg, C. Fraser, and H. Heesterbeek. The effectiveness of contact tracing in emerging epidemics. *PLoS ONE*, 1(1), 2006. [Cited on pages 202 and 204.]

[121] P. Kostkova, M. Szomszor, and C. St. Louis. #swineflu: The Use of Twitter as an Early Warning and Risk Communication Tool in the 2009 Swine Flu Pandemic. *ACM Transactions on Management Information Systems*, 5(2), 2014. [Cited on page 198.]

[122] N. W. Landry and J. G. Restrepo. The effect of heterogeneity on hypergraph contagion models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(10):103117, 2020. [Cited on pages 65 and 185.]

[123] G. Lee, M. Choe, and K. Shin. How Do Hyperedges Overlap in Real-World Hypergraphs? - Patterns, Measures, and Generators. In *Proceedings of the The Web Conference 2021*, WWW'21, pages 3396–3407. ACM / IW3C2, 2021. [Cited on page 154.]

[124] V. Leonenko, S. Arzamastsev, and G. Bobashev. Contact patterns and influenza outbreaks in Russian cities: A proof-of-concept study via agent-

based modeling. *Journal of Computational Science*, 44:101156, 2020. [Cited on page 173.]

[125] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-Effective Outbreak Detection in Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, page 420–429, New York, NY, USA, 2007. Association for Computing Machinery. [Cited on page 52.]

[126] Alexis LG. HyperGraphLib. `github.com/alex-87/HyperGraphLib`, 2016. [Online; 2021]. [Cited on page 40.]

[127] B. Li, J. Li, K. Tang, and X. Yao. Many-Objective Evolutionary Algorithms: A Survey. *ACM Computing Surveys*, 48(1), 2015. [Cited on page 71.]

[128] W. Li, Q. Bai, M. Zhang, and T. D. Nguyen. Modelling Multiple Influences Diffusion in On-Line Social Networks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'19, page 1053–1061, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems. [Cited on page 53.]

[129] Y. Li, J. Fan, Y. Wang, and K. Tan. Influence Maximization on Social Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1852–1872, 2018. [Cited on pages 50 and 51.]

[130] F. Lorig, E. Johansson, and P. Davidsson. Agent-Based Social Simulation of the Covid-19 Pandemic: A Systematic Review. *Journal of Artificial Societies and Social Simulation*, 24(3):5, 2021. [Cited on page 68.]

[131] B. Lovric and M. Hernández. The House of Black and White: Identities of Color and Power Relations in the Game Of Thrones. *Revista Nós*, 4(2):161–182, 2019. [Cited on page 107.]

[132] EPFL LTS2. gspbox. `github.com/epfl-lts2/gspbox`, 2017. [Online; 2021]. [Cited on page 39.]

[133] B. Luo, J. Zheng, J. Xie, and J. Wu. Dynamic Crowding Distance? A New Diversity Maintenance Strategy for MOEAs. In *2008 Fourth International Conference on Natural Computation*, volume 1 of *ICNC'08*, pages 580–585, 2008. [Cited on page 71.]

[134] T. Ma and J. Guo. Study on information transmission model of enterprise informal organizations based on the hypernetwork. *Chinese Journal of Physics*, 56(5):2424 – 2438, 2018. [Cited on pages 56, 66, 141, and 186.]

[135] Meta. Meta Data For Good Programme, 2022. [Cited on page 178.]

[136] M. Milkoreit. Pop-cultural Mobilization: Deploying Game of Thrones to Shift US Climate Change Politics. *International Journal of Politics, Culture, and Society*, 32(1):61–82, 2019. [Cited on page 107.]

[137] S. Moore, E. M. Hill, M. J. Tildesley, L. Dyson, and M. J. Keeling. Vaccination and non-pharmaceutical interventions for COVID-19: a mathematical modelling study. *The Lancet Infectious Diseases*, 21(6):793–802, 2021. [Cited on pages 67 and 246.]

[138] W. Muno. *"Winter Is Coming?" Game of Thrones and Realist Thinking*, pages 135–149. Springer International Publishing, Cham, 2019. [Cited on page 107.]

[139] Murali-group. halp. `github.com/Murali-group/halp`, 2014. [Online; 2021]. [Cited on pages 39 and 84.]

[140] S. Muñoz and C.A: Iglesias. An agent based simulation system for analyzing stress regulation policies at the workplace. *Journal of Computational Science*, 51:101326, 2021. [Cited on page 173.]

[141] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009. [Cited on page 71.]

[142] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004. [Cited on page 91.]

[143] BBC News. Social gatherings over six to be banned in england, 2020. [Cited on page 205.]

[144] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-Scale Networks. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 695–710, New York, NY, USA, 2016. Association for Computing Machinery. [Cited on pages 52 and 55.]

[145] C. Nowzari, V.M. Preciado, and G.J. Pappas. Analysis and control of epidemics: A survey of spreading processes on complex networks. *IEEE Control Systems Magazine*, 36(1):26–46, 2016. [Cited on page 61.]

[146] N. H. Ogden, A. Fazil, J. Arino, P. Berthiaume, D.N. Fisman, A.L. Greer, A. Ludwig, V. Ng, A.R. Tuite, P. Turgeon, and L.A. Waddell. Modelling scenarios of the epidemic of COVID-19 in Canada. *Canada Communicable Disease Report*, pages 198–204, 2020. [Cited on page 69.]

[147] R. Olesker. Chaos is a ladder: A study of identity, norms, and power transition in the Game of Thrones universe. *The British Journal of Politics and International Relations*, 0(0):1369148119885065, 2019. [Cited on page 107.]

[148] WHO World Health Organization. Non-pharmaceutical public health measures for mitigating the risk and impact of epidemic and pandemic influenza, 2019. [Cited on pages 9, 198, 199, 200, 204, and 246.]

[149] WHO World Health Organization. Water, sanitation, hygiene and waste management for the COVID-19 virus. HWO Technical Report WHO/2019-nCoV/IPC_WASH/2020.1, World Health Organization, 2020. [Cited on page 209.]

[150] P. E. Paré, C. L. Beck, and A. Nedić. Epidemic Processes Over Time-Varying Networks. *IEEE Transactions on Control of Network Systems*, 5(3):1322–1334, 2018. [Cited on page 174.]

[151] R. Patel, M.M. Raghuwanshi, and L.G. Malik. Decomposition Based Multi-objective Genetic Algorithm (DMOGA) with Opposition Based Learning. In *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, CICN'12, pages 605–610, 2012. [Cited on page 71.]

[152] S. Peng, Y. Zhou, L. Cao, S. Yu, J. Niu, and W. Jia. Influence analysis in social networks: A survey. *Journal of Network and Computer Applications*, 106:17 – 32, 2018. [Cited on pages 51 and 56.]

[153] N. Perra. Non-pharmaceutical interventions during the COVID-19 pandemic: A review. *Physics Reports*, 913:1–52, 2021. Non-pharmaceutical interventions during the COVID-19 pandemic: a review. [Cited on pages 67, 199, and 218.]

[154] T. Postmes, R. Spears, A. T. Lee, and R. J. Novak. Individuality and Social Influence in Groups: Inductive and Deductive Routes to Group Identity. *Journal of Personality and Social Psychology*, 89:747–763, 2005. [Cited on page 129.]

[155] B. Praggastis, D. Arendt, J.Y. Yun, T. Liu, A. Lumsdaine, C. Joslyn, M. Raugas, B. Kritzstein, S. Aksoy, N. Landry, E. Purvine, M. Shi, and F. Theberge. HyperNetX. `github.com/pnnl/HyperNetX`, 2019. [Online; 2021]. [Cited on pages 40, 82, 84, and 94.]

[156] H.J. Pérez and R. Reisenzein. On Jon Snow's death: Plot twist and global fandom in Game of Thrones. *Culture & Psychology*, 0(0):1354067X19845062, 2019. [Cited on page 107.]

[157] U.N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review. E*, 76, 2007. [Cited on pages 92 and 103.]

[158] R.E. Rhodes and E.P. Zehr. Fight, flight or finished: forced fitness behaviours in Game of Thrones. *British Journal of Sports Medicine*, 53(9):576–580, 2019. [Cited on page 107.]

[159] S. Rojas-Galeano and L. Alvarez. Simulation of Non-Pharmaceutical Interventions on COVID-19 with an Agent-based Model of Zonal Restraint. *medRxiv*, 2020. [Cited on page 200.]

[160] R. Rossi, V. Socci, D. Talevi, S. Mensi, C. Niolu, F. Pacitti, A. Di Marco, A. Rossi, A. Siracusano, and G. Di Lorenzo. COVID-19 Pandemic and Lockdown Measures Impact on Mental Health Among the General Population in Italy. *Frontiers in Psychiatry*, 11:790, 2020. [Cited on page 199.]

[161] T. C. Schelling. Micromotives and Macrobehavior. *The Journal of Politics*, 42(2), 1980. [Cited on page 50.]

[162] C. Senevirathna, C. Gunaratne, W. Rand, C. Jayalath, and I. Garibay. Influence Cascades: Entropy-Based Characterization of Behavioral Influence Patterns in Social Media. *Entropy*, 23(2), 2021. [Cited on page 53.]

[163] J. Shun. Hygra. `https://github.com/jshun/ligra/`, 2020. [Online; 2021]. [Cited on page 84.]

[164] J. Shun. Practical Parallel Hypergraph Algorithms. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP'20, page 232–249, New York, NY, USA, 2020. Association for Computing Machinery. [Cited on pages 39 and 42.]

[165] J. Shun and G.E. Blelloch. Ligra: A Lightweight Graph Processing Framework for Shared Memory. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '13, page 135–146, New York, NY, USA, 2013. Association for Computing Machinery. [Cited on page 39.]

[166] D. Sikeridis, I. Papapanagiotou, and M. Devetsikiotis. BLEBeacon: A Real-Subject Trial Dataset from Mobile Bluetooth Low Energy Beacons. *CoRR*, abs/1802.08782, 2018. [Cited on pages 74, 178, 210, and 213.]

[167] P.C.L. Silva, P.V.C. Batista, H.S. Lima, M.A. Alves, F.G. Guimarães, and R.C.P. Silva. COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions. *Chaos, Solitons & Fractals*, 139:110088, 2020. [Cited on pages 69 and 232.]

[168] O. Sims. networkR, R. `github.com/O1sims/networkR`, 2016. [Online; 2021]. [Cited on page 42.]

[169] N. Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.*, 2(3):221–248, 1994. [Cited on pages 71 and 232.]

[170] J. Stavanja and M. Klemen. Predicting kills in Game of Thrones using network properties, 2019. [Cited on page 105.]

[171] F. Stonedahl and U. Wilensky. Finding Forms of Flocking: Evolutionary Search in ABM Parameter-Spaces. In *Lecture Notes in Computer Science*, pages 61–75. Springer Berlin Heidelberg, 2011. [Cited on pages 70 and 71.]

[172] B. Su, P. Andelfinger, J. Kwak, D. Eckhoff, H. Cornet, G. Marinkovic, W. Cai, and A. Knoll. A passenger model for simulating boarding and alighting in spatially confined transportation scenarios. *Journal of Computational Science*, 45:101173, 2020. [Cited on page 173.]

[173] Q. Suo, J. Guo, and A. Shen. Information spreading dynamics in hyper-networks. *Physica A: Statistical Mechanics and its Applications*, 495:475 – 487, 2018. [Cited on pages 57, 64, 141, and 186.]

[174] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. [Cited on page 255.]

[175] J. Tang, Y. Chang, C. Aggarwal, and H. Liu. A Survey of Signed Network Mining in Social Media. *ACM Comput. Surv.*, 49(3), 2016. [Cited on page 4.]

[176] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1539–1554, New York, NY, USA, 2015. Association for Computing Machinery. [Cited on page 52.]

[177] E. Tekin and I. Sabuncuiglo. Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions*, 36(11):1067–1081, 2004. [Cited on pages 70 and 71.]

[178] M. Tracy, M. Cerdá, and K. M. Keyes. Agent-Based Modeling in Public Health: Current Applications and Future Directions. *Annual Review of Public Health*, 39:77–94, 2018. [Cited on pages 60 and 62.]

[179] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Pr, 1983. [Cited on page 94.]

[180] S.S. Urmy. The Julia Language is the Way of the Future. `https://www.oceanographerschoice.com/2016/03/the-julia-language-is-the-way-of-the-future/`, 2016. [Online; 2021]. [Cited on page 44.]

[181] H. Van Dyke Parunak, R. Savit, and R. L. Riolo. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In *Multi-Agent Systems and Agent-Based Simulation*, pages 10–25, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. [Cited on page 60.]

[182] F. Verelst, L. Willem, and P. Beutels. Behavioural change models for infectious disease transmission: a systematic review (2010–2015). *Journal of The Royal Society Interface*, 13(125):20160820, 2016. [Cited on page 67.]

[183] N. X. Vinh, J. Epps, and J. Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010. [Cited on page 103.]

[184] C. Wang, R. Pan, X. Wan, Y. Tan, L. Xu, R. S. McIntyre, F.N. Choo, B. Tran, R. Ho, V.K. Sharma, and C. Ho. A longitudinal study on the mental health of general population during the COVID-19 epidemic in China. *Brain, Behavior, and Immunity*, 87:40 – 48, 2020. [Cited on page 199.]

[185] R. Wang, Y. Li, S. Lin, H. Xie, Y. Xu, and J. C. S. Lui. On Modeling Influence Maximization in Social Activity Networks under General Settings. *ACM Transactions on Knowledge Discovery from Data*, 15(6), 2021. [Cited on pages 55 and 141.]

[186] Z. Wang, C.T. Bauch, S. Bhattacharyya, A. d'Onofrio, P. Manfredi, M. Perc, N. Perra, M. Salathé, and D. Zhao. Statistical physics of vaccination. *Physics Reports*, 664:1–113, 2016. Statistical physics of vaccination. [Cited on page 256.]

[187] L. Webb. COVID-19 lockdown: A perfect storm for older people's mental health. *Journal of Psychiatric and Mental Health Nursing*, 2020. [Cited on page 199.]

[188] B. Wilder, M. Charpignon, J. A. Killian, H.-C. Ou, A. Mate, S. Jabbari, A. Perrault, A. N. Desai, M. Tambe, and M. S. Majumder. Modeling between-population variation in COVID-19 dynamics in Hubei, Lombardy, and New York City. *Proceedings of the National Academy of Sciences*, 117(41):25904–25910, 2020. [Cited on pages 68 and 244.]

[189] C. Yang, X. Shi, L. Jie, and J. Han. I Know You'll Be Back: Interpretable New User Clustering and Churn Prediction on a Mobile Social Application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 914–922, New York, NY, USA, 2018. Association for Computing Machinery. [Cited on pages 190 and 213.]

[190] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in

LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015. [Cited on pages 75, 210, and 213.]

[191] Q. Yang, C. Yi, A. Vajdi, L. W. Cohnstaedt, H. Wu, X. Guo, and C.M. Scoglio. Short-term forecasts and long-term mitigation evaluations for the COVID-19 epidemic in Hubei Province, China. *Infectious Disease Modelling*, 5:563–574, 2020. [Cited on pages 69 and 232.]

[192] Yelp. Yelp. `www.yelp.it`, 2004. [Online; 2021]. [Cited on page 77.]

[193] Yelp. Yelp data set. `www.yelp.com/dataset`, 2019. [Online; 2021]. [Cited on page 78.]

[194] Yelp. Yelp data set documentation. `www.yelp.com/dataset/documentation`, 2019. [Online; 2021]. [Cited on page 78.]

[195] S. Yoon, H. Song, K. Shin, and Y. Yi. How Much and When Do We Need Higher-Order Information in Hypergraphs? A Case Study on Hyperedge Prediction. In *Proceedings of The Web Conference 2020*, WWW '20, page 2627–2633, New York, NY, USA, 2020. Association for Computing Machinery. [Cited on page 6.]

[196] Z. Zare. Representation of Gender Roles in Child and Young Characters in Game of Thrones Series. *IAU International Journal of Social Sciences*, 9(2):83–91, 2019. [Cited on page 107.]

[197] H. Zhang, S. Mishra, and M.T. Thai. Recent advances in information diffusion and influence maximization in complex social networks. *Opportunistic Mobile Social Networks*, 37(1):1 – 36, 2014. [Cited on page 51.]

[198] J. Zhu, S. Ghosh, and W. Wu. Group Influence Maximization Problem in Social Networks. *IEEE Transactions on Computational Social Systems*, 6(6):1156–1164, 2019. [Cited on pages 54, 128, and 141.]

[199] J. Zhu, S. Ghosh, J. Zhu, and W. Wu. Near-Optimal Convergent Approach for Composed Influence Maximization Problem in Social Networks. *IEEE Access*, 7:142488–142497, 2019. [Cited on pages 55 and 141.]

[200] J. Zhu, J. Zhu, J. Ghosh, W. Wu, and J. Yuan. Social Influence Maximization in Hypergraph in Social Networks. *IEEE Transactions on Network Science and Engineering*, 6(4):801–811, 2019. [Cited on pages 55 and 141.]