NETWORK BASED DATA ORIENTED METHODS FOR APPLICATION DRIVEN PROBLEMS

A DISSERTATION SUBMITTED TO THE
DOCTORAL SCHOOL OF COMPUTER SCIENCE
OF THE
UNIVERSITY OF SZEGED

by
László Hajdu

Supervisor:
Dr. Miklós Krész

Szeged, 2021

Acknowledgments

First of all, I would like to thank my supervisor, Miklós Krész for the patience, guidance, and support through this journey. We may have had disagreements about philosophical questions and the direction of this research, although I have to agree in the end, I learned a lot from him, and I am very proud and thankful that he is my supervisor.

I would also like to thank all of my co-authors, especially András Bóta from whom I learned a lot about research.

I am very grateful to my colleagues in the office (Attila, Balázs, Imi, Norbi), who inspired me even in the very early stages of my research, and made an amazing working environment, where I was happy to start every single day.

Last, but not least, I would also like to thank my parents, my sister Erika and my girlfriend Dóri for the amount of support and love they gave me from the very beginning, without them this dissertation couldn't have happened.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks are amazing. If you think about it, some of them can be found in almost every single aspect of our life from sociological, financial and biological processes to the human body. Even considering entities that are not necessarily connected to each other in a natural sense, can be connected based on real life properties, creating a whole new aspect to express knowledge. A network as a structure implies not only interesting and complex mathematical questions, but the possibility to extract hidden and additional information from real life data. The data that is one of the most valuable resources of this century. The different activities of the society and the underlying processes produces a huge amount of data, which can be available for us due to the technological knowledge and tools we have nowadays. Nevertheless, the data without the contained knowledge does not represent value, thus the main focus in the last decade is to generate or extract information and knowledge from the data. Furthermore, data analytics and science, as well as data-driven methodologies have become leading research fields both in scientific and industrial areas.

In this dissertation, the author introduces efficient algorithms to solve application oriented optimization and data analysis tasks built on network science based models. The main idea is to connect these problems along graph based approaches, from virus modelling on an existing system through understanding the spreading mechanism of an infection/influence and maximize or minimize the effect, to financial applications, such as fraud detection or cost optimization in a case of employee rostering. We introduce our work continuously where each chapter is somehow connected to the previous topic.

First, we start with an overview on network science to give a short historical introduction about the field and define concepts, definitions and terms that are needed to understand the later chapters.

In Chapter 2, we introduce a method to analyze the vulnerability of a public transportation system, using the passenger co-traveling network of the actual city. In the first part of the chapter, we present a new community definition for networks that are representing the passenger travels on public transportation, and examine the usage of the system giving frequent trip changes helping the

public transportation company to improve their service. After the passenger community detection and system analysis, we examine the different scenarios in a case of an epidemic, and detect the critical and risky vehicle trips in order to minimize the spread on the public transportation network in the city. We introduce and test our method with the public transportation network of Minneapolis. Our publication connected to this topic can be found in [78].

In Chapter 3, an interesting concept is introduced that is trying to connect the infection models and the community detection. The main idea of the chapter is based on a fact that an infection on a network spreads easier in a community so in a relatively dense subgraph than in the other parts of the network. We introduce a new methodology where on one hand, the results of different community detection models are able to improve the efficiency of the infection maximization problem, on the other hand, the infection maximization can provide a reliable benchmark system in order to rank the different community detection methods. After the concept is introduced, we test the methods from both sides on real and randomly generated networks, and give comparison on existing community detection algorithms. The methodology is studied in [76, 77]

Chapter 4, introduces an infection maximization model based on the Generalized Independent Cascade [29], using a use case from psychology. In the first part of this chapter, we focus on the new infection maximization model and a method to minimize the network effect in a case of an existing influence or infection. The second part of the chapter examines one of the possible use cases, where the objective is to improve mental wellbeing in organizations with targeted psychological interventions. The model is tested on a social network that was created based on employees of an existing nursing home in Norway. Our research connected to this chapter can be seen in [120].

Chapter 5, contains a new method to analyse temporal networks in order to detect fraud in the banking environment. The motivation of this research comes from the financial sector, where the detection of special network motifs can reveal fraud like activity in the system. The main objective of this part is to give a general view on financial fraud detection, and introduce a new cycle detection method that is able to detect special money transfer cycles in a transaction network. The method is tested on real life transaction data from 2016 and it is used in a Hungarian bank since 2018. The corresponding paper to this research can be found in [80].

In Chapter 6, we solve the crew rostering problem with a heuristic that is using a network to represent the search space of the optimization problem. The goal of our method is to minimize the cost of the employment and to create a solution that meets with the given regulations. The two step method is tested on real life problem from a public transportation company, as well as on randomly generated inputs. In the end of the chapter, we compare the results of our method with the results of a mathematical model. The chapter is studied in [81].

In Chapter 7, we conclude and summarize the dissertation in English and in Hungarian.

Table 1.1. Correspondence between thesis points, publications and chapters.

	[78]	[76]	[77]	[120]	[80]	[81]
I.	•					
II.		•	•			
III.				•		
IV.					•	
V.						•
Chapter	2.	3.	3.	4.	5.	6.

1.1 The Short History of Network Science

The history of network science and graph theory leads back to 1736, when Leonard Euler analyzed the famous bridges of Königsberg. The mathematician wanted to find a path through the city, which crosses each bridge but only once. Euler modelled the problem and created a mathematical structure, where vertices represented the different parts of the city, while the edges between them corresponded to the bridges. The term graph was introduced in 1878 by J. J. Sylvester who not only coined the word itself, but found an application in chemistry. Later in 1936, Dénes Kőnig, a Hungarian mathematician wrote the first book about the graph theory creating a completely new branch in mathematics based on Euler's original problem, that studies the structure of the different pairwise connections. The definition of the graph in the book [101] is the following: "Let A, B, C be a set of points. If certain pairs of these points are connected by one or more lines, the resulting configuration is called graph". As the mathematical definition described the graph as set of points connected by one ore more lines creating new and interesting questions in the field of mathematics, the graph theory also has proved to be a useful tool in a case of real life models. In 1933, a Romanian psychologist Jacob Moreno presented a sociogram that is a graphic representation of social connections between individuals. The network showed the interpersonal structure of an elementary school student group, and it has become the cornerstone of the field of social network analysis. Nevertheless, graph theory has shown its potential in many different fields in the early stages, such as social sciences, economical and financial areas.

Meanwhile, mathematicians started to investigate the different structural properties of the networks. Pál Erdős and Alfréd Rényi's eight papers on random graphs served as a basis for random graph theory [54] introducing the Erdős-Rényi model in order to create random graphs. In the same time, a different theroem was introduced by László Lovász [124, 123] based on the results of Tibor Gallai [66], that concentrated on strict structures called perfect graphs. In the early 90's, due to the technological development, more and more data was produced from different sources, such as internet and telecommunication companies, also it became possible to analyze the properties of real world networks. From the the small-world experiment of Stanley Milgram, it turned out that the average path length of the social network is quite low, and it induced the further analysis of real

world networks. In 1998, Watts and Strogatz published a network model [170], based on the small world theory that showed the properties of a network that describes real world connections, and just one year later, Barabási and Albert introduced the preferential attachment model [16] that is able to create scale free networks where the degree distribution follows power law.

In the later years these models provided the basics of the network science what we know today. The field started to investigate the properties of real world networks or *complex networks*, that are containing non-trivial topological features. In the last 20 years, different branches of the field were created connected to real networks, including but not limited to community detection, social network analysis, motifs, controllability, and diffusion models. The different topics of the network science that are connected to this dissertation will be explained and discussed in details.

1.2 Properties of Real World Networks

Before we introduce the main aspects and results of our research, it is important to present the basic concepts of graph theory. In this section, we introduce definitions, notations and problems that are connected to the content of this dissertation, and needed to be clarified to the reader .

1.2.1 Basic Definitions and Notations

To define an *undirected graph* or *network*, let $G(V, E)$ be an ordered pair where V is the finite set of *nodes* or *vertices*, and E is the finite set of *edges* that are unordered pairs of nodes. The vertices or nodes in a graph, so the elements of set V can be connected by edges. If $u, v \in V$ and e is an edge that connects u and v nodes, e can be denoted by $e = (u, v)$, $e_{u,v}$ or $e(u, v)$, and u and v are *neighbours* or *adjacents*. Let n_v be the set of neighbouring nodes of the v so the vertices that are connected to the v. If an edge is connected to the same vertex twice, it is called *loop*. There are different types of networks, for example if we have multiple or *parallel* edges connecting the same two nodes at least twice, the network is called *multigraph*. However, if we do not have loops, or parallel edges, we are talking about *simple graph*. Let d_v or $deg(v)$ called *degree* be the number of the edges that are connected to the node v. For a network G where the $|V| = n$, the minimal number of edges is 0 in an *empty graph* while the maximal is $n(n-1)/2$ in a *clique* or *complete graph* that is denoted by K_n. *K-clique* is a clique that has exactly k number of nodes. In a G graph, the *maximum clique* is the largest clique, so that contains the largest number of nodes, and the *maximal clique* is the clique that cannot be extended or enlarged.

In the case of a *directed* $G(V, E)$ graph, E is the set of ordered pair of nodes, so the edges have a direction associated with them. If we have the edge $e(u, v)$ that connects the u and v nodes,it is the *outgoing* or *out edge* of node u and *incoming* or *in edge* of the node v. The neigbours of a node can be divided into two groups according to the direction of the corresponding directed edge. The *out neighbours* are connected to node v by an edge that points from v. The out neighbours are

generally denoted by n_v^+. Furthermore, the other type of neighbours is the set of *in neighbours* that are denoted by n_v^- and are connected to v by an edge pointing towards the v. Consequently, d_v^- (*in-degree*) and d_v^+ (*out-degree*) can be defined to a vertex and will be the number of the incoming and outgoing edges.

The network is called *edge-labeled graph* or *vertex-labeled graph* if labels are assigned to the edges or nodes of the graph. The labels can contain additional properties or information about the given node or edge. If a number is assigned to the edges, so there is a function $f : E \to \mathbb{R}$ that maps a real number to the edges from the E, the graph is called as a *weighted graph*. The weights between the nodes can represent connection strength, distance or different properties of the connection. Since in our case most of the networks come from real life, the nodes or edges can have labels or numbers assigned.

In graph theory we use several different methodologies to represent a graph. The *adjacency matrix* is an nxn matrix denoted by A_{ij}, where the element $a_{ij} = 1$ if there is an edge between the nodes i and j, and 0 otherwise. The matrix can be symmetric (undirected graph) or assymetric (directed graph). The *edge list* is the simplest way to represent a network that contains the list of node pairs that are connected to each other. Additional properties of the edges such as weights or labels can be represented. The *adjacency list* contains the list of neighbours of each node. In this dissertation we will use edge list as a representation of networks.

A *subgraph* $S(V', E')$ of a graph $G(V, E)$ is a graph where $S(V') \subset G(V)$ and $S(E') \subset G(E)$, so a subgraph is a graph that is contained by a larger graph. In a subgraph, *restricted degree* of the node v can be defined as the number of the incident edges that are in E'. If the S contains all edges from G among the vertices of S, then S is an *induced subgraph*. Furthermore, *network motifs* are subgraphs in the network, that can be found in a larger graph repeatedly. These motifs can be defined as a pattern or different structural interactions between the subset of vertices. The important motifs will be defined in the corresponding sections of the dissertation.

A *walk* on a network is a finite or infinite alternating sequence of edges that are connecting sequence of nodes, so $e_{v_i,v_1}, e_{v_1,v_2}...e_{v_{j-1},v_j}$ is a walk that goes from v_i to v_j, however, it is called *trail* if the edges are distinct. The walk is *closed*, if the v_i equals to v_j. The length of the walk can be defined as the number of the edges on it. We are talking about a *path* if the vertices are distinct along the walk, and about a *cycle*, if it is a non-empty trail, where the first and the last node of the trail are the same and there are no repeated vertices along the cycle. *Shortest path* between v_i and v_j can be defined as a path where the length is minimal, or in a weighted network where the sum of the weights along the path is minimal.

In an undirected graph, *connected component* is a subgraph where any of the nodes are connected with each other by a path. Consequently, in directed graph, a subgraph is *strongly connected component*, if the nodes are reachable from each other (so there is a directed path between each node pairs), and *weakly connected*, if it is connected when the edges are considered undirected.

1.2.2 Community Structure in Networks

The complex or real networks often have a structural peculiarity, which implies that the nodes can be grouped into sets such that the vertices inside the set have dense connection structure. This property is called *community structure*. However, in the literature there are lot of different definitions to describe the communities, they are mostly interpreted as a dense subgraph where the nodes are more connected to each other than in the other parts of the network. To understand the real concern of the communities, the best example is the social network that describes the social connections in the society. The different community groups of a person are often formed along the activities, locations, interests, hobbies or occupation. This concept is also known as homophily in the social sciences [50, 176, 38]. Nevertheless, the community structure can be observed not only in social networks but in many other types of the complex networks.

The communities can have two different types, based on the number of the groups where a node can belong. In a case, if a vertex can be a part of only one community, it is called *non-overlapping community* or *cluster*. Consequently, if a node can belong to multiple communities, we are talking about *overlapping communities*. It is evident, that the definition and the interpretation of the communities highly depends on the real world use case.

To give a formal definition of the community, let $G = (V, E)$ be an undirected network and $S(V', E')$ be the subgraph, where $|G(V)| = n$ and $S|V'| = n_S$. We define the $d_v^{internal}$ and $d_v^{external}$ as the internal and external degree of the node v, that means the number of the edges connected to the actual node where each $e \in S(E')$ (internal degree) or $e \notin S(E')$ (external degree). Let $d_S^{internal}$ and $d_S^{external}$ be the sum of the internal and external degrees in the subgraph. Since a clique is known as the strongest community, where there is an edge between each node pairs in the subgraph and the number of the edges in a complete graph is $n(n-1)/2$, the internal density can be defined as follows:

$$\delta_{internal}(S) = \frac{d_S^{internal}}{n_s(n_s - 1)/2}$$

Furthermore, to define how dense is the given subgraph, we have to know how many edges point out from the subgraph. The maximal number of the edges that goes from S to the other parts of the network is $n_S(n - n_S)$ so the external density is:

$$\delta_{external}(S) = \frac{d_S^{external}}{n_s(n - n_s)}$$

It can be seen that the quality of the actual cluster can be the ratio of the internal and external density. In other words, the objective when we would like to define communities is to maximize the internal density and minimize the external density. However, it is important to note that the previously defined measure does not gives us a general definition, so there is no exact interpretation or method with which we can rank or describe the quality of communities. The literature gives

us different approaches to define the structure itself. The *local definition* as we have seen before concentrates on the properties of the S subgraph and on the relation of the internal nodes of the community. The *global definition* tries to give a measure on the whole graph. The modularity is defined by Girvan and Newman [136], where the idea is based on the observation, that the random graph is not expected to have community structure. The modularity is trying to measure the strength of division of the networks into groups or clusters (i.e. non-overlapping communities). The formula of the methodology can be defined as follows:

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{d_i d_j}{2m}) \delta(S_i, S_j)$$

where i and j goes through on each node pairs, the m is the number of the edges so $|G(E)|$, d_i and d_j are degrees of the node i and j. $A_{i,j}$ denotes the adjacency matrix of the graph and $\delta(S_i, S_j)$ is one if the nodes i and j are in the same community, and zero otherwise. We can say, that the modularity has proved to be a good definition describing non overlapping communities. It is possible to give a third definition based on the *node similarities*, where two nodes can belong to the same cluster if they have similar network properties. Since the networks cannot be defined in the metric space, it is possible to give a similarity measure to the nodes using the adjacency matrix. In the literature, lot of different similarities can be found, but the following measure can give us a general idea about this definition:

$$m_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - Ajk)^2}$$

The definition compares the two rows of the adjacency matrix. It is trying to express, how similar is the position of the two nodes. If they are positioned in the graph in a very similar way, that means they have similar neighbours, the vertices will be in the same community. It can be seen, that several different methods are defined in the literature in order to characterize the community structure, nevertheless the different definitions can have advantages or disadvantages if we want to define the actual structure of real world communities. The approach that we follow during this dissertation is strongly related to real life applications, therefore we examine the applicability of our methods from the actual application point of view.

Community Detection

The extraction or creation of the previously defined groups or dense subgraphs from a network is called *community detection*, and it can be a computationally difficult task. The number or the size of the communities is not known in most cases, and they often have different size or density. First we take a look about the different methodologies to find non-overlapping communities where, as it is discussed above, one node can belong only to one community.

The roots of the community detection are originated from clustering. The first graph partitioning methods were published around the 70's, such as k-means clustering [126], spectral methods [125] or

the Kernigen and Lee method [96]. In *hierarchical clustering* [133], there are two different approaches based on the methodology of the cluster creation. The first is the *agglomerative clustering*, where the nodes are in different clusters in the beginning, and we merge the groups based on the similarities of the elements. In contrast, when we use *divisive clustering* the idea is to cut the edges between the non-similar vertices, until we do not get disjoint connected components as clusters. One of the first methods for community detection was given by Girvan and Newman [70]. The method first calculates the *edge betweeness centrality* on the network, which is the number of the shortest paths that goes through an edge. Afterwards, the method removes the edges with the highest centrality until no edges remain, creating a hierarchical structure. Two years later, the same authors proposed the modularity that was discussed in the previous section, and it was proven to be an efficient stopping criteria for their method. However, the authors and other researchers proposed many different optimization techniques to create communities using the idea of the modularity such as greedy [136], extremal optimization [49] or simulated annealing [75]. These algorithms are trying to create communities that are maximizing the value of the modularity function. It has to be noted, that the modularity has its own limits, so in special cases in large networks it has the possibility to create large communities without finding more detailed resolution. Popular approaches in non overlapping community detection also include information theoretic approaches [155], statistical inference [145, 12] and spectral techniques [105, 135].

In real life networks, overlapping communities are more useful due to the fact that an individual or entity can belong to multiple groups or communities. The main problem with overlapping community structure is, that while in the case of the clustering, benchmarking the methods is relatively easier, the standard measurement of the overlapping community detection methods is a challenging task, since there is no general benchmark methodology. At the same time, it is easy to verify the results in a case of real world networks, since the communities describe real structures. However, several different approaches can be found in the literature [138]. Another methodology to compare overlapping community detection methods is to create networks with artificially created network structure [111]. The most known algorithm is the *clique percolation method (CPM)* which after detecting the k-cliques in the network, connects the cliques that have $k-1$ common nodes. The method was given by Palla et. al [140]. Although, the method is very popular and it is among the first overlapping community detection algorithms, it also has disadvantages compared to other methods. For example, the size of the initial cliques is not trivial, and since the method is detecting chain of k-cliques and not "dense" subgraphs, in the case of special structural properties, we can get unfortunate results. For example, vertices in a community can belong to non $k-1$ adjacent cliques but can be reached from another path or structure like nodes where $deg(v) = 1$. Generally it can be stated, that several other non-clique based methods can also be found in the literature for example block models [43], fuzzy clustering [178], label propagation [74] and many others [28, 113, 173]. Nevertheless, community detection algorithms can be also used in weighted networks [12].

It can be seen, that in community detection wide variety of the approaches can be considered in both overlapping or non-overlapping, weighted and unweighted case. This branch of the network science and graph theory offers extremely useful tools to know more about complex networks, and it is still a very well researched area even nowadays. A good overview on different community detection methods can be found in [59].

1.2.3 Diffusion Models

Networks and their structure can support and effectively express the modelling of different real life processes. In this section, we shortly overview the different diffusion models that can be used on networks. The area of *diffusion models* focuses on the spread or diffusion of different effects on a system and the structure of the system can be described by a network. The origin of diffusion modelling comes from epidemiological use cases, where processes can express the spread of different viruses or diseases. However, the process can come from many other areas, for example next to medical processes, we can model the spread of information, bankcruptcy, churn, or even behavior or mood between different individuals. Since an important part of this dissertation deals with epidemiological modeling on public transportation and with the dynamics of different infection models, the topic of the *diffusion processes* has to be examined. The compartmental models discretize the mathematical representation of infectious diseases in a way, that the population is grouped into different compartments based on their state. However, it is important to note, that the first diffusion models were not used on a network, but the spread was defined based on differential equations. If we consider the compartmental models, the first important model that has to be discussed is the SIR model, which was proposed by Kermack and McKendrick in 1927 [58]. In the SIR, the individuals from the population can be in three different states. The state S represents the susceptible group, which means that if an individual has infectious contact and it is susceptible, it will be transitioned into the infectious compartment. The I denotes the group of infectious individuals that can infect people from the susceptible compartment in a case of a contact. The R means removed, so they were in state I, but are healed and immune or deceased. However, many different versions of the original model were proposed in the literature. For example SIS, where the individuals do not have long immunity like in the SIR model or the SI model, where individuals can be only in the first two compartment. Another interesting model is the $SEIR$ which is similar to our first model, but there is an additional compartment that stands for the group of individuals that were exposed to the virus. The sate E means that the individual is not infectious yet, but it was infected, and will become infectious soon. The idea of the state E reflects to the incubation period of the viruses. Although, the first diffusion models were created to follow and predict the process the infectious diseases, and later they were used in graph structure, the variety of processes implied many different approaches during the last years. A good overview of general and network based epidemiological models can be found in [45, 93].

As we mentioned before, many other process or effect can spread in the real life. Granovetter was the first who proposed a model to represent the spread of different behavior patterns. His work was based on the hypothesis, that the decisions of each individual highly depends on what others are doing. Granovetter introduced the threshold model [73] based on the work of Shelling [159]. The name of the model expresses the resistance of the individual against the mass behaviour, in other words the individual will follow a behaviour if enough people from the population do the same. It has to be noted, that although the original threshold model was not defined on graphs, Granovetter suggested to use the model on social networks in his paper. Another work that has to be noted as a significant milestone of this field is the research of Domingos and Richardson, where they wanted to improve the social effects the buyers have on each other. The basic idea was to detect the most important individuals from viral marketing point of view, and spread the marketing information in the most efficient way. In this case, the idea uses the network effect to maximize the spread of the information through friend recommendations, so if we target individuals with important connections, they can recommend the product to their close connections such as friends, family etc. In their paper they also proposed several heuristics to solve the given problem.

Two years later Kempe, Kleinberg and Tardos introduced the mathematical formulation of the problem [95], creating a new infection model, the Independent Cascade (IC). In this model, the nodes can be in *active* or *inactive* state where active means that the node is infected, while inactive means the opposite. The nodes can only go from passive to active state. The Independent Cascade uses edge weights more precisely probabilities to represent the connection strength between the nodes. To define the Independent Cascade let $G(V, E)$ be a directed network where for $\forall(v, u) \in E$, there is a probability $p(v, u)$ that represents the connection strength and $0 < p(v, u) \leq 1$. The probability 1 corresponds to the strongest possible connection between two nodes, while 0 means, there is no connection between the entities. The model is iterative and starts with an initially infected node set A_0. In each iteration, the newly infected nodes try to infect their neighbours based on the edge probabilities between them. To understand the process, let us define the A_i as set of the nodes that have become infected in the i-th iteration by the set A_{i-1}. The model terminates if the set of the newly infected nodes is empty. In the end of the model, the result is the expected number of infected nodes in a case of the given A_0 initial infected set denoted by $\sigma(A)$ where $A = A_0$. The computation of the $\sigma(A)$ is #P- hard problem [36] but with simulation any precision can be reached. The work of Kempe and Kleinberg also formalizes the *infection maximization* problem based on the original work of Domingos and Richardson. The objective of this problem is to find the initial infected set so the A_0 that maximizes the expected number of infected nodes so the $\sigma(A)$. The authors proved that the problem is NP-hard [95], however, they also provided a greedy heuristic that can approximate the results with the quarantee of $1 - 1/\epsilon$. The greedy heuristic and the simulation methodologies will be discussed more detailed in the corresponding chapters to have better a understanding about the exact methods.

1.3 Algorithms and Computational Complexity

This dissertation will introduce several algorithms, so it is needed to overview the main aspects and notations of the computational complexity. In computer science, the algorithms are often analyzed or measured by the *running time* or *time complexity* that indicates the amount of time that is needed to run the actual method. The running time of an algorithm can depend on many different factors, nevertheless the time complexity is generally given with the function of the size of the input. However, since the running time can be different in a case of two different, but same sized inputs, the time complexity is measured by the number of elementary operations executed by the algorithm. Therefore, it is common to use the *worst-case time complexity* that shows the maximum time that is required for inputs with the given size. It can be stated that the computation of the exact time complexity can be challenging, even in a case of small inputs, so it is common to use the *asymptotic* function of the complexity that describes how the algorithm behaves when the size of the input increases. The most common notation of the time complexity is the big $\mathcal{O}$ notation that gives the upper bound for the running time. To define formally, let n the size of the input, $f(n) = \mathcal{O}(g(n))$ means that there are constant c and k such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$. If the time complexity of an algorithm is $\mathcal{O}(1)$, it means it is constant so only one elementary operation is needed regardless the input size. Furthermore, $\mathcal{O}(n)$ denotes linear complexity, that we get for example if we would like to find the max element in an unsorted array. Another example is the $\mathcal{O}(n!)$ so factorial time complexity that we get in a case when we want to find all permutations of a given set or string. It can be seen that algorithms with the complexity of $\mathcal{O}(1)$ or $\mathcal{O}(n)$ can be considered fast, while $\mathcal{O}(n!)$ characterizes a slow algorithm. However, the algorithm complexity is an important aspect which has to be taken into account when we are developing new methods.

1.4 Optimization and Mathematical Models

The mathematical optimization is a field where the objective is to select the best elements based on some criterion, from a set of possible configurations. Nevertheless, the elementary example is to minimize or maximize a real function choosing input values from a given set which is restricted by different conditions. Optimization can appear in many different areas such as computer science, economics, or engineering. The *objective function* defines the function that has to be minimized or maximized, and there are constraints that are defining the borders of the set that can contain the possible solutions. A solution that meets with the given constraints is called *feasible solution*, and the best available solution is called *optimum*. In the last chapter of this dissertation, an optimization problem will be introduced where we will use an integer programming model to solve our problem. In *Integer Programming (IP)*, all variables has to be integers, consequently in *Integer Linear Programming (ILP)* the objective function and the constraints are linear. The standard for of the ILP is the following:

$$\max c^T x$$
$$Ax \leq b$$
$$x \geq 0$$
$$x \text{ integer}$$

The c and b are vectors, furthermore A is a matrix and all entries are integers. From complexity theory point of view, the linear programming is NP Complete.

Since determining exact optimal solution for real-world large-scale problems with integer programming models can be problematic, heuristic methods are commonly used to search for solutions to ILP. The *heuristic* is a problem solving approach, where the objective is to find a satisfactory solution for problems that cannot be solved with exact methods. The drawback of this is, that if the method cannot find a solution, we do not know if it is because there is no feasible solution or the algorithm failed to find one. In other words, it is not known how close is the given solution from the optimum and there is no quarantee that the heuristic can find the best, or even a satisfactory solution. However, very efficient heuristic methods can be given for real life problems, and it is a common method to do experimental testing and compare the results of the heuristic solution to the results of mathematical models in a case of small problems. Basically, it can be stated that we have a trade-off from optimality, completeness, and accuracy point of view. The *metaheuristic* is a higher-level procedure where the objective is to find a heuristic that can solve the optimization problem. The metaheuristic methods can be separated based on the general idea on which they are built. According to this, we can speak about *local/global search* methods e.g. tabu search or variable neighbourhood search, *nature inspired methods*, such as ant colony optimization or particle swarm optimization, hybrid or probabilistic based algorithm and many other types of methods. A good overview on different metaheuristics can be found in [18]. In general, optimization and mathematical models have several different approaches and techniques, however, the previously given methods were shortly introduced, since they are strongly connected to some parts of the dissertation.

Chapter 2

Community Detection and Infection Modelling on Public Transportation Networks

Networks can be used to represent public transportation systems from various unique perspectives. Traditionally nodes and edges represent the physical infrastructure of a transportation system, e.g. routes and stops/stations [122], while recent developments in data collection and modeling allow researchers to accurately map contacts between individuals traveling together. Detailed commuting patterns can be recorded from smart card data [163, 164, 15] or can be the output of activity-based travel models [23, 161, 132, 35, 68, 156]. The resulting passenger contact patterns are both spatial and temporal in nature, and include travel times on specific vehicles and contact with other travelers [150, 89, 100]. Such detailed travel patterns can then be used for various planning objectives including the estimation of the capacity of infrastructure [169], calculating environmental impact [30] or designing surveillance and containment strategies during an epidemic outbreak [147, 151]. While these methods allow researchers to map contacts between known individuals, the data collection and processing required to recreate a real-world contact network presents many challenges in terms of accuracy and computational complexity, among other issues such as privacy [88, 87, 14, 158, 65, 134].

In this chapter, we propose the development of two novel network structures, namely the *transfer network* and the *community network*. The transfer network captures the movement patterns of atomic passenger groups, i.e. groups of people who travel together on one or more vehicles, mathematically defined as maximal complete subgraphs of the contact network. The community network, that captures the similarity of travel patterns between passengers, is derived directly from the transfer network and is constructed using a novel link-based metric called the *connection strength*. The community network is intended to reveal groups of travelers who may also be more likely to come

into contact outside of their daily travel routine e.g.: colleagues traveling to work or children going to school [176, 38]. A description of each proposed network structure, as well as the contact network can be found in Table 1.

We further demonstrate potential applications of each of these novel network structures. First, the transfer network is implemented to detect the most frequent vehicle trip combinations in the system (bus and train transfers). While there are existing methods in the literature to measure the capacity of public transit systems [32, 109], our approach provides a more refined view of passenger movement between vehicle trips by tracking the movements of groups of passengers traveling together. Identifying the most relevant vehicle trip combinations can aid public transit authorities in timetable planning and optimizing vehicle assignments.

Second, the community network is applied to evaluate a diffusion process in a transit network, specifically infectious disease spread among passengers. This application complements the previous work of Bóta et. al [27], to identify the components of the public transportation system most vulnerable to a bio-security threat. The community network proposed here can further improve the performance of such models due to its novel representation of passenger interactions. We use the proposed network structure to simulate how a disease might spread among the vehicles of the public transportation system and the suburbs of the city, and identify the vehicle trips most likely to carry infected passengers. All applications are evaluated on a real-world case study, the public transit system of Twin Cities, MN, using output from an activity based travel demand model [100].

The section 2.1 introduces the travel demand model, and how it is used to create passenger contact network. Afterwards in the section 2.2, we introduce the transfer network, and illustrate how it can be used to detect frequent vehicle trip combinations. The section 2.3 introduces the community network, including the connection strength value, and illustrates how it can be used to model infectious disease spread. The section 2.4 and 2.5 contains the limitations of this research, our conclusions and future research directions.

Table 2.1. Network definitions used in this chapter

	Contact network	Transfer network	Community network
Nodes	Passengers	Atomic passenger groups	Passengers traveling on at least two vehicle trips
Edges	Passengers are connected if they are physically present on the same vehicle trip at the same time (undirected)	Atomic passenger groups are connected if they share a passenger (directed)	Passengers are connected if they are traveling together on at least two vehicle trips (undirected)
Attributes	Contact duration and start time	Number of transfer passengers between atomic groups	Connection strength measuring the similarity of the travel patterns between passengers

2.1 Model Inputs

The inputs of our work are based on the transit assignment model published in [100]. In this section, we present a description of the assignment model and provide a short summary of the properties of the passenger contact network built from it.

2.1.1 Travel Demand Model

As described in the work of Bóta et. al [27], public transportation data in this study was obtained from the transit system in Twin Cities region in Minnesota, where 187 routes serve 13,700 stops in the region. Transit network and schedule data were created from General Transit Feed Specification (GTFS), including near 0.5 Million stop-times on a weekday in 2015. Transit passenger trips were obtained from Metropolitan Councils's activity-based demand model [90], and contained more than 293,000 linked trips (i.e. a passenger trip from an origin to a destination that may include zero or more transfers). The assignment of transit demand to transit network was done using the FAST-TrIPs model [100]. FAST-TrIPs is a schedule-based transit assignment model that generates hyperpaths using a defined logit route choice model [98], assigns individual passengers to the paths using the hyperpath probabilities, and simulates them using a mesoscopic transit passenger simulation module. Since a calibrated transit route choice model was not available for the Twin Cities network, a route choice model from Austin, TX was used from [99]. The model specifies the following route choice utility function:

$$u = t_{IV} + 1.77t_{WT} + 3.93t_{WK} + 47.73X_{TR}$$

where $t, t_{IV}, t_{WT}, t_{WK}$ and X_{TR} represent path utility, in-vehicle time, waiting time, walking time, and number of transfers in a transit path, respectively. The output of the transit assignment model contains individual passengers' trajectories, including their walking from the origin to the transit stop, boarding the transit vehicle, alighting and walking to the transfer stop, boarding the next transit vehicle, etc. and finally alighting and walking to the destination. By post-processing the transit assignment model's outputs, the amount of time each pair of passengers are on-board the same transit vehicle were calculated on a daily basis.

2.1.2 Contact Network

We define a network structure denoted as the *contact network* based on the outputs of the travel demand model. The nodes of the contact network are passengers, and edges connect passengers if they were traveling on the same vehicle at the same time. The relationship is symmetric, therefore the network is undirected. All passenger movements take place in a temporal setting, which is indicated by two values assigned to the edges of the network: the contact start time on an edge indicates

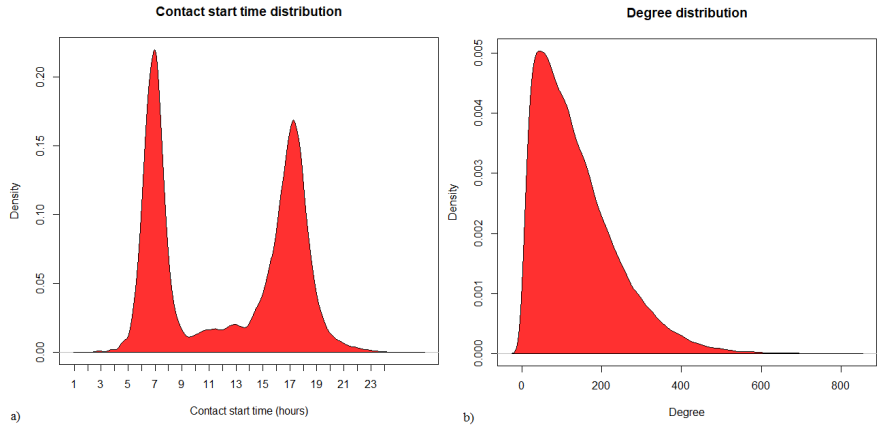


Figure 2.1. The distribution of a) contact start times and b) the degree distribution of the contact network.

the start time of the contact between the two connected passengers, while a contact duration value indicates the length of the contact in minutes. Since each edge represents a connection between a pair of passengers traveling on a specific vehicle, the id of the vehicle can also be assigned to the edges of the graph. The vehicle id identifies a vehicle trip: a single vehicle traveling on a specific route with a specific start time. The basic properties of the network were shown in [27] along with a detailed analysis regarding the networks potential role in the spreading of epidemics.

The contact network corresponding to the Twin Cities dataset has 94475 nodes and 6287847 contacts between them. Figure 2.1/a shows the density plot of the contact start times. The distribution has two peaks, one around 7 AM and another around 5 PM, which corresponds to the morning and evening weekday commute. The average number of contacts per person is 136 during the observed day, while the maximum is 827. Figure 2.1/b shows the degree distribution of the graph.

Figure 2.2 shows a subgraph of the network structure. Nodes represent passengers and edges connect them if they ride on the same vehicle trip. The dynamics of the network are omitted in this example, i.e. edges are aggregated over the entire day. The black node in the middle represents a passenger with a high number of contacts, who traveled together with all the other nodes shown on this subgraph. The other nodes are colored according to the vehicle trip they rode on, while darker edges indicate longer contact durations.

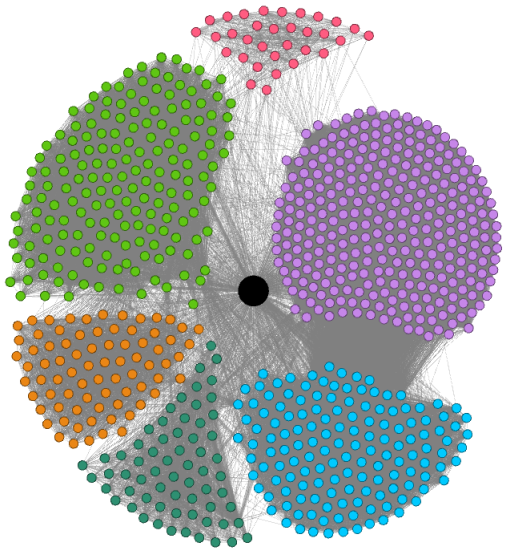


Figure 2.2. A static subgraph of the contact network. A passenger with a high number contacts is represented by the black node in the middle, connected to all other contacts. The colors indicate of the vehicle trips the passengers first met on. The figure was made with Gephi using the Fruchterman-Reingold algorithm.

2.2 Transfer Network

The first objective of this chapter is to detect the movement patterns of passenger groups. To do this, we define a novel network structure denoted as the *transfer network*. The transfer network is a directed network, where nodes represent the atomic passenger groups of the contact network and groups are connected if at least one member of a group transfers from one vehicle to another. The weight of the edges denote the number of transfer passengers.

In order to build the transfer network we identify the subgraph corresponding to each vehicle trip, detect atomic passenger groups – defined as maximal cliques – on each of the resulting subgraphs of the contact network and connect the atomic passengers groups according to direction of transfer between vehicle trips. We weight the connections based on the number of transferring passengers. The transfer network proposed in this chapter also provides a refined way to identify the most frequent vehicle trip combinations in the public transit system, which can aid decision makers in defining timetables and optimizing vehicle assignments.

In the following section, we define the construction of the transfer network in more detail, and discuss the application for detecting frequent vehicle trip combinations.

2.2.1 Transfer Network Construction

The three steps required to build the transfer network are discussed in the subsequent subsections. First, we define how subgraphs, corresponding to the vehicle trips are constructed, then we introduces the atomic passenger groups as cliques and show how they can be detected in an efficient way and in the end of this section we introduce how the transfer network can be built from the passenger groups.

Graph Partitioning

We define the subgraph corresponding to each vehicle trip as follows. Let G be the original contact network, $V(G)$ the vertices and $E(G)$ the edges of the network. We divide the original network into subgraphs along vehicle trips. Let T be the set of all vehicle trips in the network and let $t_i \in T$ denote the i-th trip. We define G_{t_i} as the subgraph corresponding to trip t_i where, $V(G_{t_i})$ and $E(G_{t_i})$ are the vertices and the edges of trip G_{t_i}. We create a subgraph G_{t_i} for all trips $t_i \in T$. Since a single passenger may travel on multiple trips, the node corresponding to the passenger may appear in multiple subgraphs. Figure 3 shows an example of the partitioning process.

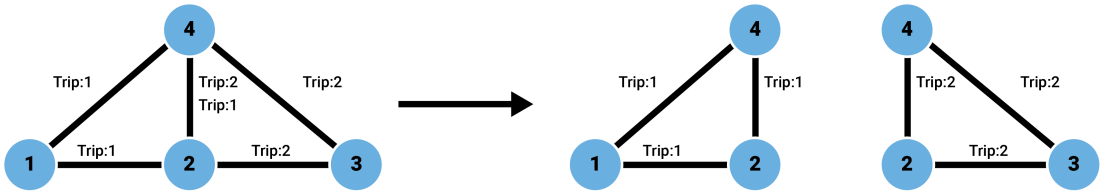


Figure 2.3. Partitioning of an example graph along vehicle trips. Each vehicle trip has a corresponding subgraph where nodes are passengers who used the given vehicle trip.

Clique Detection

In graph theory, a *clique* is defined as a fully connected subgraph of a given graph. A *maximal clique* is a clique, that is not a subgraph of any other clique. Finding the set of all maximal cliques is a well-studied NP-hard problem in graph theory. An arbitrary n-vertex graph may have up to $3^{n/3}$ maximal cliques, but this number is much lower in many complex networks, including the contact network studied in this chapter of the dissertation. Among the available methods in clique detection we adopt the Bron-Kerbosch (BK) algorithm [24], which has proven its reliability in real-life applications [53]. We apply the BK algorithm[1] to detect the maximal cliques of all subgraphs G_{t_i} corresponding to all vehicle trips $t_i \in T$.

[1]We implemented the Bron-Kerbosch algorithm with pivoting and degeneracy ordering in the outer most level of recursion in C++. We ran the algorithm on a PC with I7 4790 CPU (3.6 Ghz) and 16GB of RAM.

The analysis can be further extended if we consider the graphs as weighted according to the contact durations available on the edges. Introducing a weight threshold τ we can prune the edges of the graphs by omitting all edges with contact durations below τ. This allows us to redefine what a connection means in the network: passengers are only considered to be connected if they spend at least a certain amount of time on the same vehicle. We define three thresholds to represent the strength of connection between passengers $\tau_5 = 5$ minutes, $\tau_{15} = 15$ minutes and $\tau_{30} = 30$ minutes. We chose these thresholds to represent short, medium and long duration contacts between passengers. We prune the edges of the contact network by these values resulting in graphs G_5, G_{15} and G_{30}. In additional, let G_0 and τ_0 represent the original (uncut) graphs and the corresponding threshold. We run the Bron-Kerbosch clique detection algorithm on these graphs and analyze and compare results. The speed of the detection algorithm is amplified by the fact, that all subgraphs corresponding to vehicle trips are interval graphs. In order to show this speedup, we also run the clique detection algorithm on the original contact network and compare results. Table 2.2 shows graph size, runtime of the clique detection method on the original contact network and the total runtime of the detection method on all subgraphs for G_0, G_5, G_{15} and G_{30}. We can see a significant speedup for all graphs in this analysis. For the larger G_0 and G_5 graphs the total runtime on all subgraphs corresponding to vehicle trips is 14 times less than on the unpartitioned network.

Table 2.2. Runtimes of the Bron-Kerbosch on graphs G_0, G_5, G_{15}, G_{30}. Raw denotes runtime in seconds on the original contact network, while partitioned denotes the total runtime on all subgraphs corresponding to vehicle trips.

Graph	Passengers	Edges	Trips	Time raw (s)	Time partitioned (s)
G_0	94475	6435482	8002	5285	367
G_5	91894	5203557	7934	3830	258
G_{15}	63714	1630522	6969	1195	50
G_{30}	26154	218233	4083	129	6

Graph Building

Let F denote the *transfer network* as a directed graph where every node $v \in V(F)$ is a clique from G_{t_i} for all t_i. Edges connect nodes v and u if the corresponding cliques do not lie in the same vehicle trip, yet they have at least one common passenger. More formally, for nodes u and v in the transfer network and their corresponding cliques c_v and c_u in the contact network, u and v is connected if $c_v \cap c_u \geq 1$ and if $c_v \subseteq G_{t_v}, c_u \subseteq G_{t_u}$ then $G_{t_v} \neq G_{t_u}$.

The *direction of edges* correspond to the direction of the transfer between t_u and t_v, that is whether the passengers of $c_v \cap c_u$ move from t_u to t_v or the opposite. We establish the direction by looking at the contact start times for the individuals in $c_v \cap c_u$ in the following way. For all edges of the transfer network $e_{uv} \in E(F)$, let c_{uv} denote the set of corresponding passengers in G

as $c_{uv} = c_u \cap c_v, c_v \subseteq G_{t_v}, c_u \subseteq G_{t_u}$, $G_{t_v} \neq G_{t_u}$. Let $\alpha_{xy}^{t_i}$ denote the contact start time between passengers x and y on vehicle trip t_i. For all passengers $p, q \in c_{uv}$, if $min(\alpha_{pq}^{t_u}) < min(\alpha_{pq}^{t_v})$, then the direction of the edge $e_{uv} \in E(F)$ is from u to v, else it is from v to u. If $|c_v \cap c_u| = 1$, then let $c_v \cap c_u = \{p_0\}$ and $p \in c_u \setminus \{p_0\}, q \in c_v \setminus \{p_0\}$. Then $min_p(\alpha_{p_0 p}^{t_u}) < min_q(\alpha_{p_0 q}^{t_v})$ decides the direction of the edge as above. We assign integer values $w_v = |c_v|$ to all $v \in V(F)$ and $w_{u,v} = |c_v \cap c_u|$ for all $e(u, v) \in E(F)$. Values w_v and $w_{u,v}$ represent the amount of passengers corresponding to both the nodes and edges of the transfer network.

2.2.2 Detecting Frequent Vehicle Trip Combinations

The *transfer network* allows us to identify the most frequent vehicle trip combinations passengers travel on in the public transportation system. Detecting the most frequent combinations helps decision makers in defining timetables and optimizing vehicle assignments.

As before, let T be the set of vehicle trips, and $t_i \in T$ the i-th vehicle trip. We define vehicle trip pairs as follows: for all t_i and t_j, t_{ij} is a vehicle trip pair where $i \neq j$ and $t_i, t_j \in T$, while the set of all vehicle trip pairs is denoted by T^p. We assign a number m_{ij} to each vehicle trip pair $t_{ij} \in T^p$ indicating the amount of passenger traffic between the corresponding trips. To calculate this value let the set $V(C_{t_{ij}})$ contain all nodes of the transfer network corresponding to all cliques $c_{t_{ij}} \in C_{t_{ij}}$ where $c_{t_{ij}} \in G_{t_i} \cup G_{t_j}$, and take the subgraph induced by $V(C_{t_{ij}})$. The number m_{ij} is the sum of all edge weights of the induced subgraph. This approach offers a more refined view of passenger traffic between vehicle trips because it represents the movements of passenger groups as opposed the behavior of individual passengers.

To give another dimension to our analysis we compared the frequent vehicle trip combinations in both G_0, G_5, G_{15} and G_{30}, that is only considering passengers to be in contact with each other if they travel together for more than $\tau_5 = 5$ minutes, $\tau_{15} = 15$ minutes and $\tau_{30} = 30$ minutes in addition to the unfiltered network G_0. Table 2.3 shows the five most frequent vehicle trip combinations t_{ij} and the amount m_{ij} of passenger traffic between them.

Almost all trip combinations in Table 2.3 follow a similar pattern. Results for G_0 and G_5 are nearly identical. The most frequent combinations for G_{15} connect two additional suburbs (Oakdale and Stillwater) to G_0 and G_5, but otherwise are identical. The vehicle trips pairs with the greatest amount of passenger traffic between them are mostly in the morning or afternoon peak hours. Almost all of the trip combinations link one of the outlying suburbs to the city center, indicating the daily commuting patterns of workers or students. Just the urban area of Twin Cities covers 2646 km^2 with a population of more than three million. This means that commuters trying to reach the city center from one of the outlying suburbs must travel on two or sometimes three separate routes to get to their destination. The pattern – also shown on Figure 2.4 – is the following. Commuters start the journey from one of the *smaller outlying suburbs* like Ridgedale (route 614), Inver Grove and other suburbs south of St. Paul (route 68) or Oakdale and Stillwater (route 294). Then they change

Table 2.3. The five most frequent vehicle trip combinations in G_0, G_5, G_{15} and G_{30}. The first number indicates the route number followed by the start time of the sepcific vehicle trip.

Graph	t_i	t_j	m_{ij}	Graph	t_i	t_j	m_{ij}
G_0 1.	614/5:25	675/5:30	38	G_{15} 1.	68/6:19	94/7:04	21
2.	68/5:46	94/6:25	35	2.	94/18:32	68/18:52	19
3.	68/6:19	94/7:04	27	3.	68/5:46	94/6:24	18
4.	901/6:18	415/7:02	25	4.	614/5:16	675/5:35	16
5.	415/17:12	901/17:33	23	5.	294/5:38	94/6:44	15
G_5 1.	614/5:25	675/5:30	37	G_{30} 1.	19/17:22	850/18:03	8
2.	68/5:46	94/6:25	35	2.	54/18:02	68/18:23	4
3.	68/6:19	94/7:04	27	3.	71/6:07	61/6:38	4
4.	901/6:18	415/7:02	25	4.	5/0:30	901/2:07	4
5.	415/17:12	901/17:33	23	5.	902/19:25	68/19:52	4

vehicles in the transport hub of one of the *major outlying suburbs* (St. Paul, Minnetonka, Mall of America in Bloomington), and take an express service to the *city center* (routes 94, 675, 850, etc..).

While the frequent combinations for G_{30} are similar, there are differences. While in the suburbs these route connections are different, we can see almost the same patterns as before: travel from one of the smaller outlying suburb to a major one and then to the city centre (combinations 71–61 and 5–901). One specific route (850) is one of the longest express bus route in the city and it includes a long section where the bus doesn't stop at all. One exception to the pattern is route 54, which connects St. Paul International airport to St. Paul and route 68 connecting to south St. Paul. In terms of time in addition to peak hours, we see late night services as well.

We can summarize, that graphs G_0, G_5 and G_{15} behave similarly, showing the movements of passenger groups traveling through the public transit system. The most frequent trip combinations are the ones connecting distant suburbs to the city center during the morning and afternoon peak hours. G_{30} captures an alternative set of routes corresponding to people who travel together for longer periods for different reasons, like a long service (route 850), the scarcity of services late at night (route 5 at 0:30) or traveling from the airport to a major transport hub (route 54).

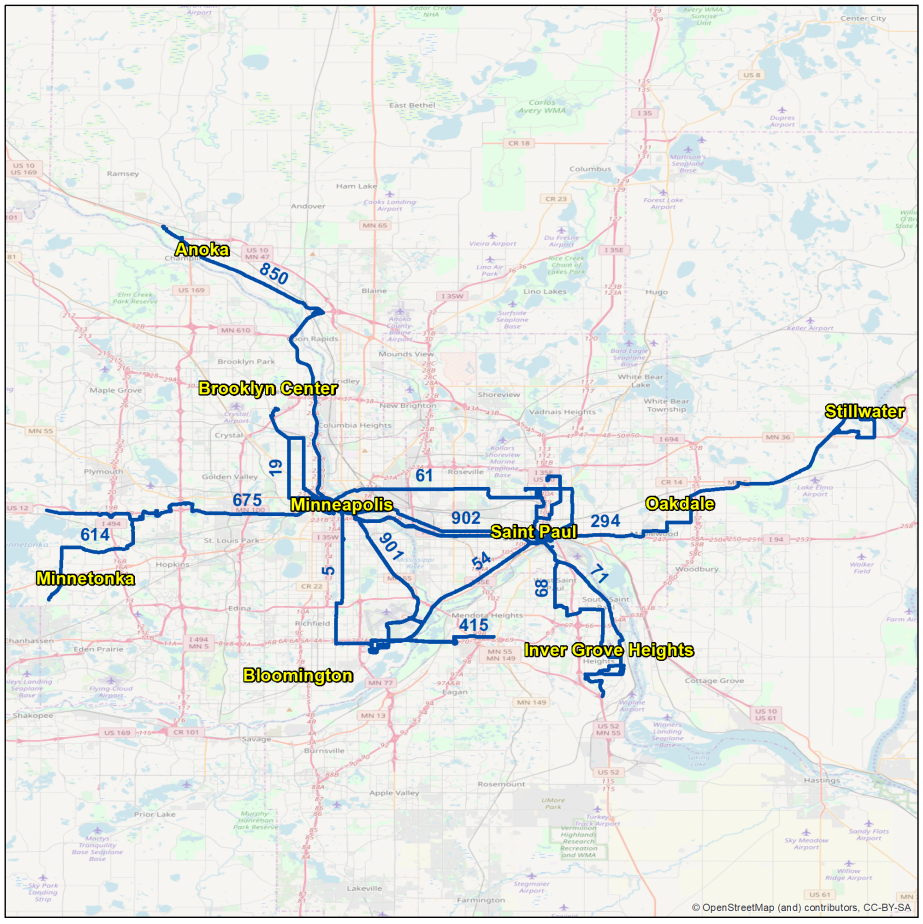


Figure 2.4. Most frequent vehicle trips combinations in Twin Cities, MN for G_0, G_5, G_{15} and G_{30}.

2.3 Community Network

Next we propose a novel network structure, the *community network*, which expands the definition of passenger connectivity to be a function of both the number of transfers passengers make together as well as the total amount of time they spend together while traveling. This contrasts the contact network, which simply quantifies passenger connectivity passengers using contact duration on individual vehicle trips. We define a *community of passengers* as a set of passengers who have common travel patterns, e.g., vehicle trips and/or transfers. In order to build communities, we define and quantify a novel *connection strength* metric between passengers indicating the similarity of their travel patterns and create a new network structure, the *community network* based on this value. Section 2.3.1 outlines this process in more detail.

This network can serve as the basis of a community detection algorithm, but in this chapter we take a different approach. We define communities as those connected by edges whose values lie

above a predetermined threshold. In section 2.3.2 we demonstrate this feature by identifying the commuting patterns of the members of the largest passenger community of the network.

We propose an application of the community network in section 2.3.3. We show how the connection strength value can be used to model infectious disease transmission among passengers of a public transit system. We also examine the different scenarios and identify the critical parts of the public transportation system in a case of an outbreak .

2.3.1 Community Network Construction

In order to detect the communities of the public transportation network we construct a weighted network structure called *community network*. The community network connects passengers using on a novel link-based metric, the *connection strength*. The connection strength s defines the edge weights in the community network, and takes into account the number of transfers a pair of passengers makes together. Thus, if the passengers meet on multiple different vehicle trips, their connection strength s will increase. This method is based on the assumption that passengers who not only travel together but also transfer together have a stronger connection than travelers who are simply present on the same vehicle trip at the same time. Below we explain how this link metric is derived.

Let H denote the new network where the nodes are the passengers, and the set of nodes V(H) includes all passengers who traveled on at least two vehicle trips with another passenger. Thus, the nodes in the community network correspond to the edges of the transfer network. We define the connections and weights between passengers as follows. Nodes u and v in the community network are connected if they are both present in at least two different cliques c_1, c_2 in two different subgraphs corresponding to vehicle trips, *i.e.* they traveled together on at least two vehicle-trips ($u, v \in c_1 \subseteq G_{t_1}$ and $u, v \in c_2 \subseteq G_{t_2}$). Let g_{uv} denote the number of instances where u and v are members of the same clique, that is $g_{uv} = |C_{uv}|$ where $c_{uv} \in C_{uv}$ if $u, v \in c_{uv}$. Let T_{uv} be the set of *vehicle trips* where both u and v are present: $t_{uv} \in T_{uv}$ if $u, v \in t_{uv}$, and let $g_{uv}^{t_i}$ be the number of the cliques in vehicle trip t_i where u and v are both present: $g_{uv}^{t_i} = |C_{uv}^{t_i}|$ where $c_{uv}^{t_i} \in C_{uv}^{t_i}$ if $u, v \in c_{uv}^{t_i} \in G_{t_i}$. Thus, the connection strength s_{uv} between passengers u and v can be formalized as follows:

$$s_{uv} = \frac{g_{uv} * (g_{uv} - 1)}{2} - \sum_{t_i \ in \ T_{uv}} \frac{g_{uv}^{t_i} * (g_{uv}^{t_i} - 1)}{2} \tag{2.1}$$

Using this definition of connection strength, if a pair of passengers traveled together in g_{uv} different atomic passenger groups, then they would have $\frac{g_{uv}*(g_{uv}-1)}{2}$ different edges between them because the affected nodes form a clique in the transfer network. This way the first part of the equation rewards the movements between different vehicle trips. Since based on our community definition traveling on the same vehicle trip does not indicate strong connection between the passengers, in the second part of the equation we penalize any instance where u and v travels together on the same

vehicle trip. The value of the penalty will be the sum of the edges in every vehicle trip where the passenger pair appears more than one time, and the number of the edges in a vehicle trip is counted in the same way as in the first part of the equation.

Algorithm 1 Construction of the *community network*

```
1: V(H) ← Every passengers from E(F) sections
2: E(H) ← ∅
3: For in E(F)sections Do
4:     For u v in every passenger pairs Do
5:         If e(u, v) ∉ E(H)
6:             E(H) ← e(u, v)
7:         Else
8:             s_uv + +
9:         End if
10:    End for
11: End for
```

Algorithm 1 shows the construction of the *community network*, while Figure 2.5 illustrates a few examples for computing s between passengers. On Figure 2.5/a two passengers travel together in two cliques on vehicle trip t_1 and one clique on vehicle trip t_2, therefore $g_{uv}^{t_1} = 2,\ g_{uv}^{t_2} = 1,\ g_{uv} = 3$ and $s = 2$. A different situation is shown on 2.5/b where two passengers travel together in three cliques on t_3 and two cliques on t_4 making $g_{uv}^{t_3} = 3,\ g_{uv}^{t_4} = 2,\ g_{uv} = 5$ and $s = 6$. Figure 2.5/c presents a trivial case, when two passengers travel together on a single vehicle trip in four cliques making $g_{uv}^{t_1} = 4,\ g_{uv} = 4$ and $s = 0$.

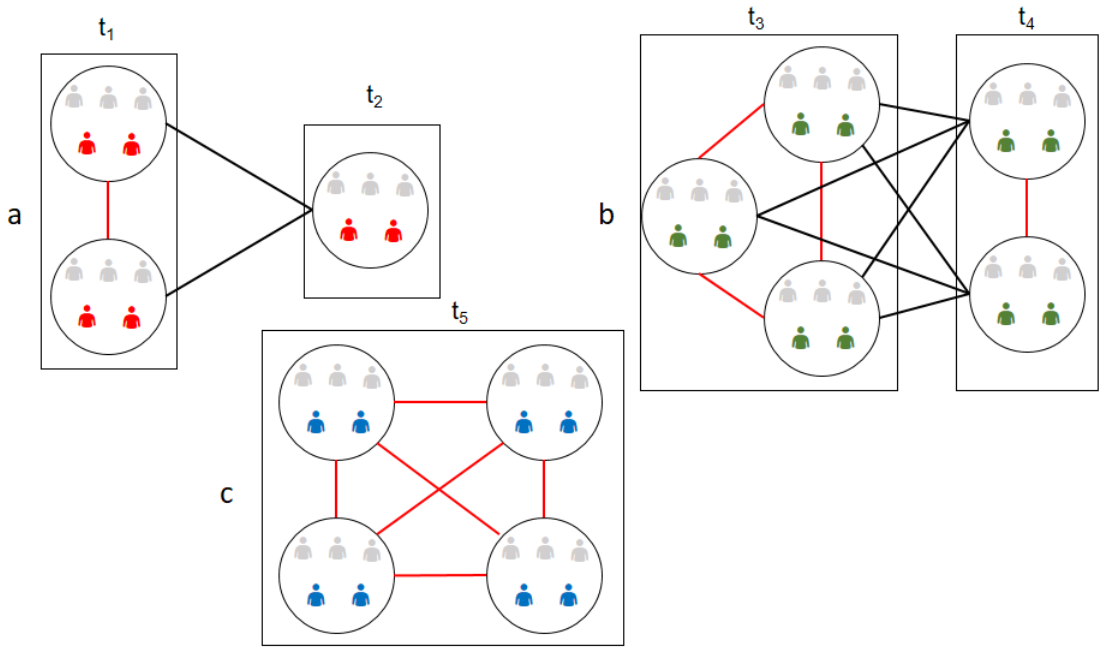


Figure 2.5. Examples of the connection strength between pairs of passengers in three different travel scenarios.. Rectangles represent vehicle trips and circles represent cliques. Edges marked with black increase the connection strength between highlighted passengers. Red edges penalize connection strength because these are on the same vehicle trip. Figure 2.5/a: two passengers travel together in two cliques on vehicle trip t_1 and one clique on vehicle trip t_2, therefore $g_{uv}^{t_1} = 2,\ g_{uv}^{t_2} = 1,\ g_{uv} = 3$ and $s = 2$. Figure 2.5/b: two passengers travel together in three cliques on t_3 and two cliques on t_4 making $g_{uv}^{t_3} = 3,\ g_{uv}^{t_4} = 2,\ g_{uv} = 5$ and $s = 6$. Figure 2.5/c: two passengers travel together on a single vehicle trip in four cliques making $g_{uv}^{t_1} = 4,\ g_{uv} = 4$ and $s = 0$.

2.3.2 Passenger Communities

In this section we illustrate examples of passenger communities in the public transportation system of Twin Cities MN. The first example seen on Figure 2.6 shows subgraphs of the community network constructed from G_{30}, *i.e.* the contact network only containing edges where contact duration is above 30 minutes. Figure 2.6/a depicts the entire community network. Most of the communities on this network are of size two or three, but there are several larger communities with strong connections between the members. Figure 2.6/b shows a subgraph where edges with weights $s < 5$ are omitted as well as all nodes with degrees below two. The remaining subgraph contains the largest group of the network, while the largest individual community is depicted on 2.6/c.

Figure 2.6/c shows the largest group in the network. The group contains nine passengers who traveled together on two different vehicle trips, while the overall time they spent using the public transportation network was almost 1.5 hours. The passengers embarked on route 805 in the morning between 7:08 and 7:16 near Blaine and traveled together to Northtown. They disembarked at 7:48

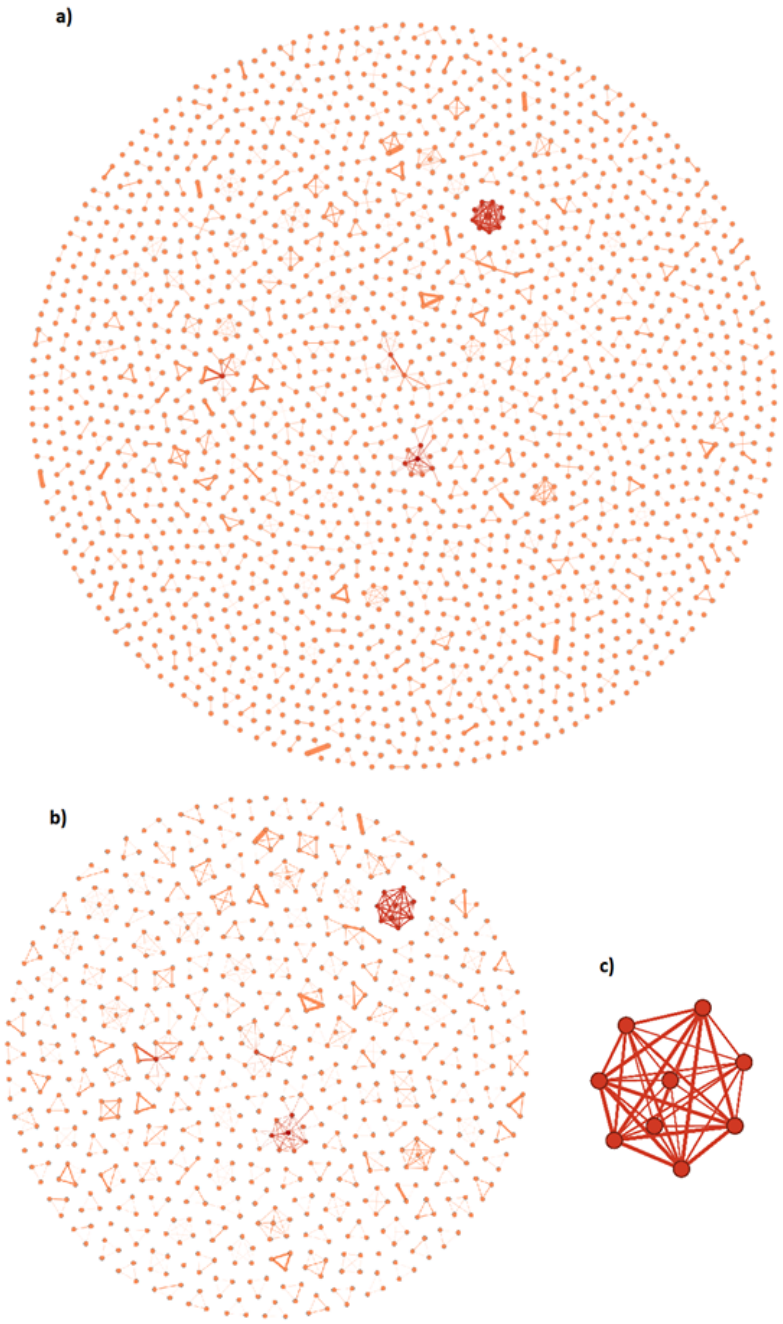


Figure 2.6. The community network of Twin Cities, MN. a) the whole community network, b) a subgraph with edge weights greater than 5, c) the largest passenger group of the network.

and waited together for the second vehicle 852 arriving at 8:12 and traveled together to downtown Minneapolis for almost an hour until 9:12 and 9:16. The travel path of the community, shown on Figure 2.7, indicates a commuting pattern from one of the suburbs to the city center of Minneapolis.

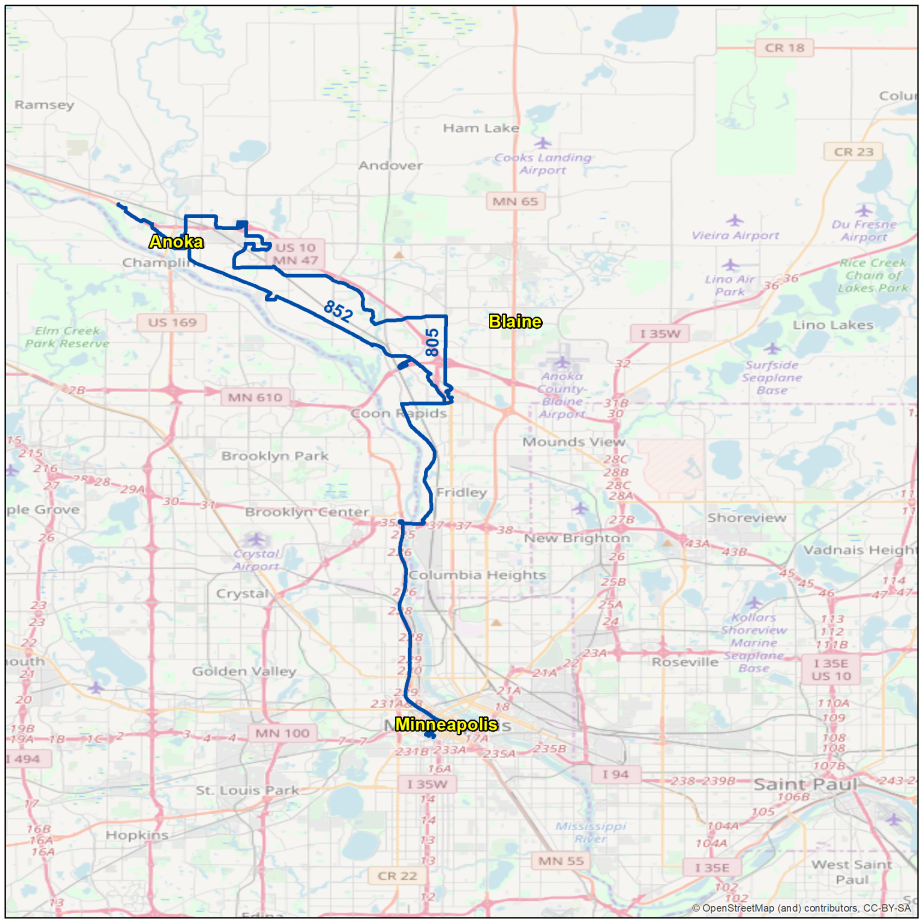


Figure 2.7. The travel path of the passenger community on Figure 6/c traveling from a suburb to the city center.

Both contact and community networks reveal contact patterns between passengers of a public transportation system. In the contact network, the weight between individual passengers is defined based on the contact duration on individual vehicle trips. In contrast, the community network provides a more refined way to represent connection strength which takes into account the amount of transfers passengers take together.

A we discussed in the first chapter of the dissertation, the underlying concept behind community structure in networks is homophily, which is a well-studied concept of the social sciences [50, 176, 38]. Homophily states that people tend to form groups according to their lines of interest, occupation, etc. Physical proximity – on public transportation for example – is one of these indicators. Therefore,

strong connections on the community network can help us uncover connections in other areas of life like workplace or school or other common interests. It should be noted, that physical proximity does not guarantee another type of connection, it simply increases the likelihood of occurrence for it.

2.3.3 Epidemic Spreading Risk Application

One application of identifying the communities within the transit network, as described in this chapter, is infrastructure security. Understanding passenger communities enables more efficient and accurate tracking of infectious disease spread, were one to be naturally or maliciously introduced into the public transit system. One of the challenges in modeling epidemic spreading is accurately mapping the relationships between individuals traveling on the same vehicle. A traditional contact network as defined in the first section of this chapter as well as in [27, 164, 26] simply reveals the set of passengers who were present on the same vehicle and the amount of time they spend on the same vehicle. As shown in [26], this limits the options for how infection spreading probabilities can be defined to be simply a function of contact duration, without the possibility to take into account physical proximity, communication etc. between people.

In contrast, the weights of the community network reveals a deeper level of connections between travelers, as passengers with strong connections in this network may also be connected in other areas of life. Passenger groups identified in the community network are more likely to be traveling within close physical proximity of each other and interacting with each other [50, 176, 38]. As a consequence, the probablity of disease transmission between the travelers belonging to the same community is greater than between two passengers simply sharing the same vehicle without any other connection. This is especially true for large public transportation vehicles like trains or trams, where simply being present on the same vehicle may not imply any kind of connection at all.

In this section we expand the previous work of Bóta et. al. [27] which examined epidemic spreading risk in the same public transportation network (Twin Cities, MN) with the goal of identifying the vehicle trips most likely to carry infected passengers. The analysis was presented in two parts. First, the passenger contact network, in the same form as in fist section of the chapter, was used to model a variety of outbreak scenarios. The scenarios differed in the number and distribution of initially infected passengers and the level of infectiousness, represented as the risk of spreading between passengers. The spreading risk was defined as the contact duration multiplied by a constant value. The scenarios were compared in terms of their impact on the network and confirmed previous observations in the literature, that is that the most central vehicle trips in the public transportation network are also the most susceptible to infection. The second part of the analysis focused on a newly proposed vehicle trip network, which represents the public transit network as a network of vehicle trips instead of passengers. It was shown by the authors that centrality metrics on the vehicle trip network provided a more efficient way to detect the set of vehicle trips most susceptible to disease, and this estimation can be done much faster than running simulations on the contact

network.

In the rest of this section we present an alternative way to model the risk of disease spreading between passengers, which exploits the community network structure introduced in this work. We define the infection transmission probability between a pair of passengers to be a function of both the connection strength value used in the community network and contact duration. We believe this multidimensional transmission probability more accurately represents the level of connectivity between pairs of passengers. Using these new transmission functions, we implement similar spreading scenarios to the ones in [27] and identify and rank the vehicle trips susceptible to disease spreading. Below we define the new transmission probability function and the infection model, then present the new vehicle-trips identified to be at highest risk.

Experiment Setup

In order to simulate an epidemic outbreak on the contact network we use the well-known discrete compartmental susceptible-infected (SI) model. As a reminder of the corresponding section of the first chapter, let us define the model in more detail. In the SI infection process all nodes adopt one of two available states: susceptible (S) or infected (I). Real values denoted as edge infection probabilities are assigned to the links of the network and denoted as $w_e \in [0, 1]$. The infection spreads in a network when susceptible nodes adopt an infected state. This is done in discrete time steps in an iterative manner starting from an initially infected set of nodes. In each iteration each infected node tries to infect its susceptible neighbors according to the transmission probability of the link connecting them. If the attempt is successful, the susceptible node is transformed into an infected one in the next iteration. In this work we limit the number of iterations to five, representing a complete work week with recurrent commuting patterns. The inputs of this model are: 1. a contact network of individuals, 2. an assignment of weights to the links of the network which represent the infection transmission probabilities and 3. the set of initially infected nodes, e.g., individuals.

We use the original structure of the contact network, which connects all pairs of passengers that travel on a vehicle-trip together as the basis of this experiment. The link weights are computed in the following way. Since the nodes and links of the community network are a subset of the nodes and links of the contact network, if a link between two nodes is present in the community network we will assign its connection strength value to the corresponding link in the contact network. We do this in all cases where such an assignment can be made. In order to account for extreme outliers of connection strength we cap all values at 100, then rescale all values to the interval of 0.1 and 0.8 using a standard feature scaling method. The 0.8 upper bound is set because a transmission probability of 1.0 is assumed to be unrealistic. For all links of the contact network that do not have a corresponding link in the community network we assign a uniform infection value of 0.05. In contrast to the duration-based value used in [27], this enables us to capture the increased spreading risk between passengers sharing similar travel patterns, while still allowing disease to spread between

travelers simply sharing a vehicle. Future work will explore the model's sensitivity to the uniform infection assignment using in this work.

Following the procedure in [27], we randomly select 100 passengers from the network to be initially infected. Due to the probabilistic nature of the simulation model, we run the SI infection model $k = 10000$ times to quantify the likelihood of each node being in an infectious state at the end of the fifth time step.

Vehicle Trip Ranking

The infection model constructed above provides the likelihood of infection for each passenger in the contact network at the end of the simulation, i.e., after five days. As in [27], we compute a similar infection value for vehicle trips by summing the probability of infection for all passengers on a given vehicle trip. While this does not represent a probability value in a strict sense, this value is proportional to the risk of getting infected on a given vehicle trip.

Table 2.4. Ranking of vehicle trips where infection is most likely to appear.

Rank	Route number	Start time
1.	25	6:52
2.	17	6:13
3.	25	5:11
4.	14	16:54
5.	5	5:36
6.	25	7:16
7.	18	6:07
8.	11	15:52
9.	11	17:14
10.	61	6:07

The routes which contain the highest risk vehicle trips, and corresponding travel times identified in this study are presented in Table 2.4. Figure 2.8 shows the routes on the map of the city. Coinciding with the findings in Section 2 of this chapter and also in [27], the vehicle trips are in the morning and afternoon peak hours. Also coinciding with the results in [27] all of the high risk routes identified go through the city center.

Since the goal of Section 2.2.2 – identifying the most frequent trip combinations – and the goal of this section – identifying the most risky vehicle trips in the case of an epidemic outbreak – are different, the difference between the selected vehicle trips are not surprising. We observe some similarities between the "highest risk" vehicle trips identified here, and the most frequent vehicle trip combinations (identified in section 2.2.2). For example, route 18 connects Bloomington with the city center, while route 61 connects St. Paul to the same destination. We have seen in Section 2.2.2, that these target-destination pairs are present in the most frequent combinations as well. As

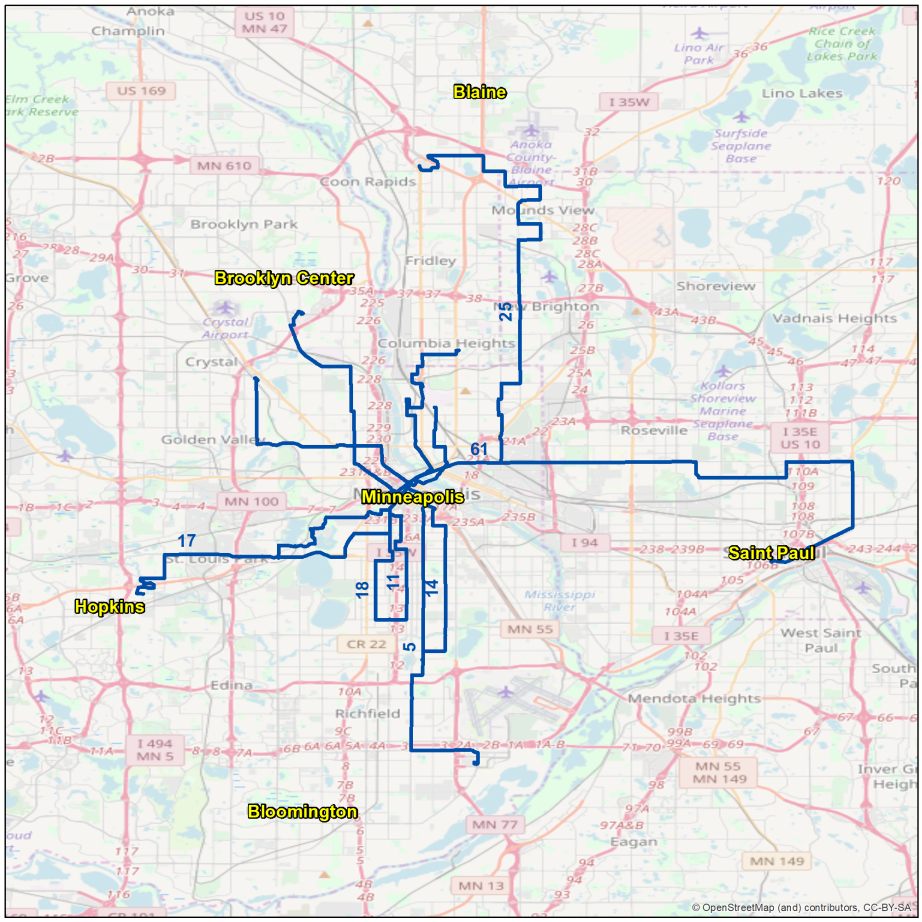


Figure 2.8. Vehicle trips most likely to carry infected passengers in the public transit system of Twin Cities, MN.

these routes connect the most important parts of the city, some similarity is expected.

The set of high risk vehicle trips identified here partially correspond to those identified in [27]. Specifically, the methodology proposed in [27] identifies vehicle trips on the routes 7, 9, 11, 14, 17, 25, 61 and 94 as most at risk. These are routes crossing or connecting to the city center in the peak hours, so even though they do not completely match those in Table 2.4, the pattern they present is the same. The similarity in the set and type of routes identified in both studies points to the critical role of the network structure in modeling outbreak risk. Future research will continue to build on this application, and further explore the robustness and sensitivity of the proposed methodology. The relevant sensitivity analysis is, however, outside the scope of this work.

2.4 Limitations

This study is subject to certain limitations. We note, that the transit demand model used as an input of both works was used to generate commuting patterns for single workday. This limitation is most critical for the epidemic application due to the implicit assumption that daily travel patterns remain constant during a five day work week. While this assumption is supported by a recent study [164], more long term observations potentially involving weekends and public holidays would improve the quality of the results presented in this chapter of the dissertation.

Further, while we present an alternative metric to quantify disease transmission risk between passengers (compared to contact duration alone), without any real-world observations on an outbreak validating the results of this work remains a challenge, and will be the focus of future research. One solution would be to use epidemic data recorded in another major city [163], but adapting such data sources to different environment presents its own set of challenges.

2.5 Summary

In order to better understand passenger commuting habits in a public transportation system, in this chapter we analyzed the contact patterns from a public transit assignment model in a major metropolitan city. We did this by defining two novel network structures to detect and quantify the movement patterns of passengers. The transfer network tracks the movements of atomic passenger groups while the community network links passengers with similar travel patterns. We presented applications for both networks, specifically to identify the most frequently used travel paths, i.e., routes and transfers, and model epidemic risk posed by passengers of a public transit network, respectively.

Our main findings correspond to existing literature, while providing a more detailed analysis of passenger commuting habits. We have shown that the most frequent vehicle trip combinations follow a similar pattern. The trip pairs identified using the transfer network identified peak morning and afternoon trips that connect the outlying suburbs with the CBD. The transfer network was demonstrated as a tool to efficiently detect the most frequently used vehicle trip combinations in the public transportation system.

Our results also reinforce previous observations in revealing components of the transit system at risk from an epidemic outbreak. For this purpose, we have shown how the connection strength value can be used to estimate physical contact between passengers and to identify the vehicle trips most likely to carry infected passengers: the routes crossing or connecting to the city center in the peak hours.

Chapter 3

Infection and Communities

The general community structure allows us to discover detailed information about different groups of a network. In other words, the communities can reveal several additional and hidden properties of an individual or a group. As it is discussed in the introduction section of this dissertation, community detection is a well researched area of network science and a large variety of methods exists in the literature [59]. However, it can be seen, that validating the results of an arbitrary community detection method, especially in an application-oriented way, remains an open problem.

The Independent Cascade [95] model, as it is discussed in the introduction, provides a possible scenario of how an actual spreading event can happen. The inputs of this model are: a graph, an assignment of edge infection probabilities to its edges and the set of initially active vertices. The process is iterative and in each iteration, every active vertex tries to activate its neighbors with the probability assigned to the edge connecting them. In the same paper the influence maximization problem is defined where we are looking for the initial vertex set which maximizes the expected number of the infected vertices. While the optimization problem is NP-complete, Kempe et al. proposed a greedy method that gives a guaranteed precision result. Nevertheless, the problem with the previously mentioned method is its running time which can be unacceptable in a case of real size networks.

Connected to these two topics, it can be observed that the communities, or dense subgroups in the society can play a critical role in the spreading of viruses or the diffusion of information in real life. In other words, if we have dense connection structure, the virus or information has higher probability to spread between our entities. Since the structure of the communities and the diffusion models are naturally connected, the produced information from both can be used to improve their efficiency, which leads us to the main idea of this part of the dissertation.

In this chapter of the dissertation, we use the connection between community detection and diffusion models as our main aspect and present new community based infection maximization method, which can be also used as a benchmarking system in order to rank different community detection

methods. In the first part of the section, we introduce the basic idea of the methodology and we rank eight well known undirected community detection algorithms from the literature. Afterwards, a directed version of an undirected community detection method is introduced to reduce the search space of the original infection maximization problem.

3.1 Independent Cascade Model

As we showed before in the introduction chapter, there are numerous models of infection spreading in the literature, and these models were adopted to many different scientific fields including epidemics, sociology and economics. The infection model that we examine and use in this chapter is the Independent Cascade Model (IC). It is important to understand the process of an infection, so the rest of this section describes this model in detail.

To define the IC formally, let $G = (V, E)$ be an undirected or directed network, where $\forall(v, u) \in E$ edge has a $p(v, u)$ probability where $0 < p(v, u) \leq 1$. The nodes in the network can be in a susceptible, infected (activated) or in a removed state, corresponding to the states of the SIR compartmental model with an infectious period of one iteration. The spreading process starts from a set of initially infected nodes $A_0 \in V(G)$, and takes place in an iterative way in discrete steps. Let the set of active nodes in iteration t be denoted as A_t. In each iteration, infected nodes $v \in A_t$ try to infect their susceptible neighbours $u \in V(G)/\bigcup_{i=0\ldots t} A_i$ based on the edge probabilities. If the attempt is successful, u joins the set of infected nodes A_{t+1} in the following iteration. If more than one node is trying to infect u in the same iteration, the attempts are made independently of each other in an arbitrary order within the same iteration. The process terminates naturally at iteration t' when $A'_t = \emptyset$. Algorithm 2 summarizes the Independent Cascade Model.

Algorithm 2 Independent Cascade

```
Let A_0 denote the set of initially infected nodes
While A_i ≠ ∅
    A_i ← newly infected nodes
    ∀v ∈ A_i tries to infect their neighbors with p(v, u)
   If the infection is successful
      A_{i+1} = A_{i+1} ∪ u
   End If
End While
```

Let $\sigma(A_0)$ denote the expected number of infected nodes with A_0 as the initial set. Let $w_f(v)$ be the final infection probability of a given node. The value of the $\sigma(A_0)$ formally is the following:

$$\sigma(A_0) = \sum_{v \ in \ G(V)} w_f(v) \tag{3.1}$$

There are numerous examples in the literature to compute the $\sigma(A_0)$ [29, 162]. However, the

exact computation of $\sigma(A_0)$ is a #P-Complete problem [37].

3.1.1 Complete Simulation

In this chapter, the complete simulation algorithm proposed in [29, 95] is used to compute the expected number of infected nodes. A generalized version of the model can be found in [29]. In Complete Simulation algorithm sample size is an important parameter because it sets the number of independent simulations, such as the precision of the result. The Complete Simulation algorithm for the Independent Cascade Model is shown on Algorithm 3.

Algorithm 3 Complete Simulation

1: **Input:** Graph G, sample size s
2: $A_0 \leftarrow$ initially infected nodes
3: $j \leftarrow 0$
4: $\forall v \in G(V) : f_v = 0$
5: **While** $j < s$
6: $\quad \forall e \in G(E)$ let the edge active or passive based on $p(e)$
7: $\quad$ Modified DFS from $\forall v \in A_0$
8: $\quad$ **If** $n \in G(V)$ node is accessible from $v \in A_0$
9: $\quad\quad n : f_v \leftarrow f_v + 1$
10: $\quad$ **End If**
11: $j \leftarrow j + 1$
12: **End While**
13: $\forall v \in G(V) : fv \leftarrow \frac{fv}{s}$

The simulation generates s different instances of the original network, each having different, randomized edge infection probabilities, and in every independent simulation each edge is either in an active or a passive state. The modified Depth First Search uses only active edges to visit the nodes, and increases the f_v values of the nodes if they are visited in the simulation instance. Finally, the f_v values are divided by the number of the independent simulations, which gives us an expected value for every node. The complete simulation is used to get the $\sigma(A_0)$ value for a given initially infected set.

3.1.2 Infection Maximization

The influence maximization problem can be defined in the following way. As before, let $A_0 \in V(G)$ be the set of the initially infected nodes, let k be the cardinality of A_0 and let $\sigma(A_0)$ be the corresponding expected size of $\bigcup_{i=0\ldots t'} A_i$, containing all nodes infected during the process. The objective is to maximize the number of activated nodes on the network, when choosing k initial infectors.

To try different varieties of these sets, several repeated computations of the simulation is needed. If we want to try all possible initially infected sets for $k = 2$ of the example on Figure 3.1, we need

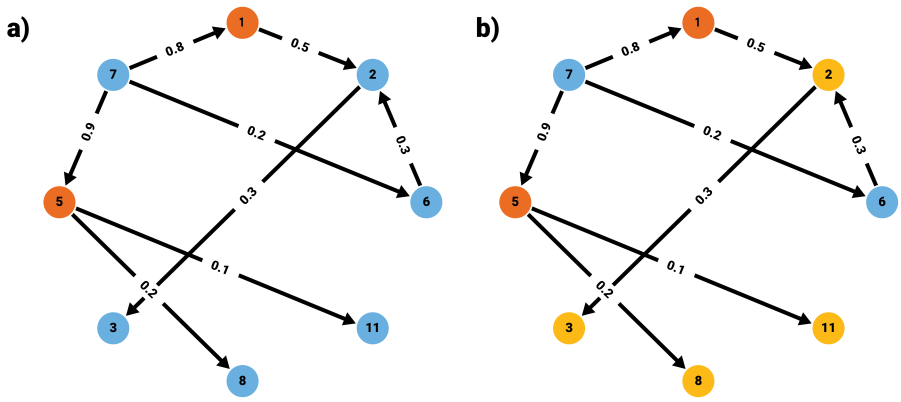


Figure 3.1. On figure a) the $A_0 = \{1, 5\}$ so nodes 1 and 5 are infected initially and $k = 2$. The figure b) shows the result of the simulation with samplesize of 100 000. The orange nodes are the initially infected nodes, the yellow nodes have greater infection than zero, and the green nodes are uninfected. In this example $\sigma(A_0) = 2.94546$.

56 different simulations for this small network, but in a real-sized network it is not computable in acceptable time. The original problem was introduced by Kempe et. al in [95] where they also have proven the NP-hardness of the problem and introduced a greedy optimization method which provides at least 63% of the optimum. In the next section we introduce the greedy algorithm.

3.1.3 Greedy Method

The greedy method starts from an empty set and increases the number of the initially infected nodes until it reaches the given k. In every iteration the algorithm chooses the node that currently seems to be the best choice. The algorithm does not give the optimal solution but it has a guaranteed precision of 63% of the optimum. However, based on our experiences, it gives much better solution in real life cases. First, the algorithm chooses the most infectious node from the network which can maximize the spread alone in the most efficient way. After that in every iteration one node is added to A_0 which gives the greatest improvement of the spread of infection with the other selected nodes. In the end, the algorithm gives an infected node set which maximizes the expected value of the infection. Algorithm 4 shows the greedy method.

Algorithm 4 Greedy method

1: **Input:** Graph G, size of the infected set k
2: **Output:** A_0 infected set
3: $A_0 \leftarrow \emptyset$
4: **While** $|A_0| \leq k$
5: $\quad A_0 = A_0 \cup \ arg\ max_{v \in G(V) \setminus A_0)} \sigma(A_0 \cup \{v\})$

In every iteration of the algorithm the next node is chosen from a $\{G(V) \setminus A_0\}$ set. Let us take an example where in the first iteration the greedy algorithm has chosen the node v_1. Afterwards, in the second iteration the method searches the second node v_2 from $G(V) \setminus A_0$ set. It can be seen that in the first iteration it computes σ so the expected value of the infection in $|G(V)|$ different cases, while in the second iteration in $|G(V)| - 1$ different cases, since v_1 is already in A_0 and v_2 will be chosen from the remaining nodes. The algorithm follows the same logic iteratively until the size of A_0 reaches the k. It can be seen that the algorithm itself is very computational intensive since it has to check the whole search space in every iteration. The idea of the chapter is to reduce the set of possible nodes minimizing the search space of the greedy method in every iteration based on some computed value which comes from the results of a community detection.

3.2 Reduction Methods

The idea of our methodology is based on an assumption that the nodes can be ordered based on their position in the network. Using only the highly ranked nodes in the greedy algorithm can significantly reduce the search space and the running time. Let $G(V*) \subset G(V)$ be a reduced node set, where the greedy algorithm chooses from $G(V*) \setminus A_0$ in every iteration. Let $f(v) : v \to Z$ be a function that assigns an integer number to every node. The nodes are ordered based on their $f(v)$ value, and the nodes with the highest $f(v)$ scores can be included in the set $G(V*)$. In this section different reduction methods are demonstrated based on a computed value assigned to every node describing the quality of a node as an infector.

3.2.1 Community Value

The communities are dense, strongly connected subgraphs where the nodes have stronger connection with each other than with the other parts of the network. If the subgraphs are dense enough, infection or influence can spread between the nodes more easily. Nodes connecting different communities in multiple dense subgraphs play a critical role, that can be used for influence maximization. Let $f_c(v) : v \to Z$ be a function for the community value that assigns to each node the number of communities it belongs to. Before we run the greedy algorithm we order nodes of the network based on their f_c value. We select the top X percent of the nodes from this ordering to become reduced $G(V*)$ set, which will replace (and reduce) the search space of the original greedy algorithm. Figure 3.2 shows an example of the community value.

Considering only the individual communities, the red nodes and the green nodes are the same because they are just simple members of the group. However, analyzing the nodes globally, the green nodes can be less effective in disease spreading because they are only connected to nodes inside their communities while the red nodes have access to both communities. In order to understand the methodology in detail, Algorithm 5. shows our reduction method based on the community value.

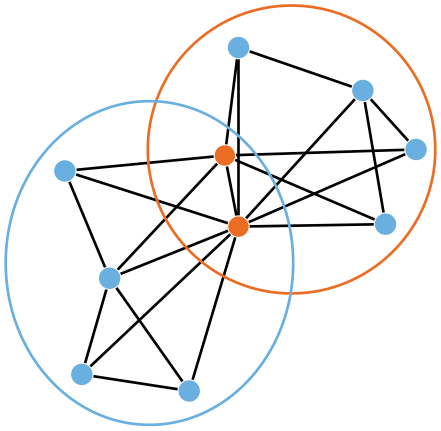


Figure 3.2. Community values in overlapping communities. Since two different communities contain the orange nodes, the community value of the red nodes is 2. The community value of the green node is 1.

Algorithm 5 Reduction method based on community value

1: **Input:** Graph $G(V, E)$, $k = |A_0|$, $X = |V^*(G)|$, M overlapping community detection method
2: **Output:** A_0
3: Detect communities on $G(V, E)$ using M
4: Ordering the nodes in set V based on their $f_c(v)$ values
5: $V^*(G) \leftarrow$ top X nodes from the ordered list
6: $A_0 \leftarrow \emptyset$
7: **While** $|A_0| \leq k$
8: $\quad A_0 = A_0 \cup \ arg\ max_{v \in V^*(G) \setminus A_0)} \sigma(A_0 \cup \{v\})$

The algorithm takes the $G(V, E)$ network, the desired size of the A_0, the desired size of the reduced selection set $V^*(G)$ and an overlapping community detection method M as parameters. In step 3-5, the reduced $V^*(G)$ set is created based on X and the given community detection method. Step 6-8 is the optimization part where the greedy method chooses the next node from $V^*(G)$ in each iteration. Naturally, the community value can be used both in weighted and unweighted networks. In an unweighted case the community value only denotes the number of the corresponding communities, while in a weighted case the community value can be weighted by the mean of the weights on the edges. However during the testing our objective is to examine the behaviour of our methodology in unweighted cases.

Simplified heuristic

While reducing the size of the selection set reduces the runtime of the greedy algorithm, it still needs to compute the optimization steps 7-8 of Algorithm 5. In this part we show that the optimization steps can be completely omitted, while still achieving better performance than the original greedy

method. Algorithm 6 shows the simplified heuristic. Instead of setting the size of a reduced selection set to a fraction of the nodes of the network, we simply select the top k highest ranking nodes according to f_c and return them as the output A_0.

Algorithm 6 Simplified algorithm.

1: **Input:** Graph $G(V, E)$, $k = |A_0|$, M overlapping community detection method
2: **Output:** A_0
3: Detect communities on $G(V, E)$ using M
4: Ordering the nodes in set V based on their $f_c(v)$ values
5: $V^*(G) \leftarrow$ top k nodes from the ordered list
6: $A_0 \leftarrow V^*(G)$

3.2.2 Selected Overlapping Community Detection Methods

The idea of the community value is based on an output of an arbitrary overlapping community detection algorithm. In order to identify which detection algorithm matches the problem of influence maximization and our methodology best, we selected eight methods among the most popular approaches. An important part of the selection criteria was that the method had to have a publicly available implementation.

- One of the first overlapping community detection methods proposed was the Clique Percolation Method (CPM) of Palla et al. [140]. It is based on a percolation process defined between k-cliques. It is still frequently used and has a publicly available implementation online [1].
- The COPRA algorithm [74] proposed by S. Gregory, is a label propagation method for overlapping community detection based on the non-overlapping algorithm of Raghavan, Albert and Kumara [149]. An implementation of the method can be downloaded from [2].
- The Greedy Clique Expansion method [116] uses the maximal cliques of the input network and expands them by greedily maximizing a local fitness function based on the function defined by Lancichinetti et al. [112]. The algorithm is available for download at [3][1].
- The map equation forms the core of the well-known Infomap community detection method [155]. It models the probability flow of random walks on a network as an information flow and seeks to compress the description of this flow. The method was extended for overlapping communities [57] and is available for researchers on [5].
- The MOSES algorithm [128] was proposed to detect highly overlapping community structure. The algorithm uses a variant of Overlapping Stochastic Block Modeling to define a global objective function and employs a greedy maximization strategy to assign nodes to communities. It can be downloaded from [6].

[1] Unfortunately the method it is no longer available on this website.

- Another frequently used detection algorithm is OSLOM [114]. The method provides great flexibility, it is able to detect directed, weighted, overlapping and hierarchical communities, and can also be used for the refinement of existing community structures. Like many other approaches, it is based on the local optimization of a unique fitness function. OSLOM can be downloaded from [7].

- Stochastic block modeling, or more specifically, inferring the underlying stochastic block model behind the communities of an existing network is a popular and well-studied approach [46, 146]. Stochastic block models can be defined for overlapping community structure too. In our work we have used the software framework available from [4].

- The SLPA method [174] is a label propagation approach closely related to COPRA. It employs a speaker-listener information propagation process, allowing nodes to switch between the roles of speaker and listener during the stochastic detection process. During this process the nodes accumulate knowledge of repeatedly observed labels. The algorithm is available online from [8].

3.2.3 Benchmark Networks

Benchmark networks were created using the C implementation of the graph generator proposed by Andrea Lancichinetti and Santo Fortunato in [111]. The following parameters were used during the graph generation:

- N: 1000 (number of nodes)
- d: 25 (average degree)
- d_{max}: 33 (maximum degree)
- μ: $0.1, 0.2, \ldots, 0.6$ (mixing parameter)
- t_1: -2 (minus exponent for the degree sequence)
- t_2: 1.5 (minus exponent for the community size distribution)
- c_{min}: 10 (minimum for the community sizes)
- c_{max}: 50 (maximum for the community sizes)
- o_n: $0.1, 0.2, \ldots, 0.6$ (number of overlapping nodes)
- o_m: $2, 3, 4$ (number of memberships of the overlapping nodes)

The parameters were chosen so the test networks contain a large variety of different community structures. Six different values were chosen for parameter o_n governing the fraction of overlapping nodes and three values for o_m setting the number of communities a node may belong to. Additionally, six different values were chosen for the mixing parameter μ influencing the amount of connections inside and between the communities. Since the graph generator is stochastic, 10 test networks where generated with each parameter configuration resulting in $6 * 3 * 6 * 10 = 1080$ test graphs.

3.3 Comparison of Undirected Community Detection Methods

3.3.1 Original Heuristic Algorithm

We conduct the evaluation of the original heuristic in the following framework. We start by applying all community detection algorithms in section 3.2.2 to the test networks described in section 3.2.3 Then we assign community values to the nodes of the test networks for all detection methods. Let $f_c^a(v_j)$ denote the community value of node $v_j \in V(G_j)$, where G_j is the j-th test network and a marks the algorithm. We rank the nodes according their community values for each algorithm and each test graph. We also define the set of influence maximization tasks, one for each test network G_j, by setting $k = 50$, that is we are looking for the 50 most influential nodes. Then we run the heuristic algorithm on all test graphs 8 times, each time using community values obtained with one of the eight community detection algorithms. We initially set the size of the reduced selection set to 20%.

Our results that can be seen on Figure 3.3 show, that on 873 out of 1080 test networks the heuristic is able to provide better $\sigma(A_0)$ than the greedy algorithm. The heuristic performs better on test graphs with a heavily overlapping community structure, indicated by greater values of parameters o_m and o_n. Since our algorithm is based on the overlaps between communities, this behavior is not surprising. The mixing parameter μ also has an effect on the quality of the results, with greater values improving performance.

On test networks with moderately or highly overlapping community structure, the community values provided by the overlapping Infomap algorithm provides the best results in almost all situations, outperforming the other approaches and the greedy algorithm on 732 test graphs out of 1080. The only exception to this pattern is when $\mu = 0.1$ and 0.2 with a low amount of overlaps. In these situations CPM is able to challenge the greedy algorithm in a reasonable amount of cases, although the greedy algorithm provides the best result in the majority of the scenarios. SLPA occasionally gives the best result if $\mu = 0.1$, while SBM occasionally preforms best without any discernable structure. Table 3.1 shows the number of times the community values provided by a detection algorithm provides the best performance, while Figure 3.3 depicts the performance of the heuristic with

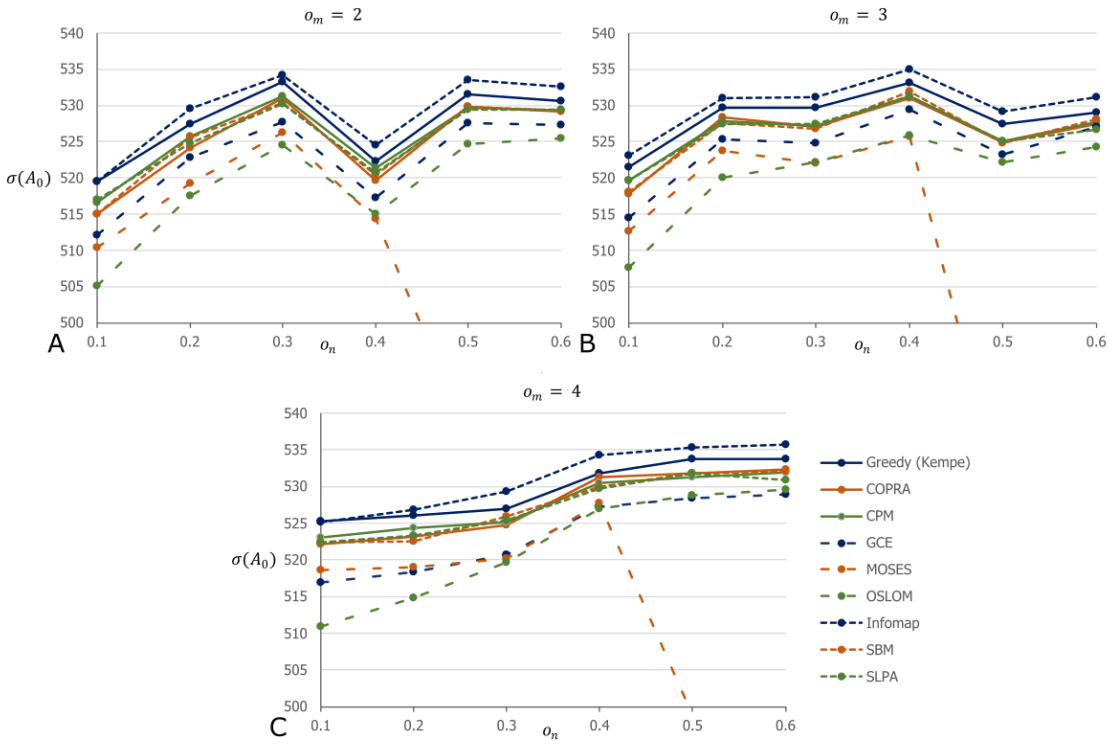


Figure 3.3. Performance of the community detection algorithms in conjunction with the heuristic with the selection set reduced to 20%. $\sigma(A_0)$ is shown for all algorithms with varying values for o_m and o_n, and $\mu = 0.3$. The performance of the greedy algorithm is also shown for comparison.

$\mu = 0.3$ for varying values of o_n and o_m. On Figures 3.3 and 3.4 results for test graphs with the same parameters (see 3.2.3) were averaged.

To further extend our analysis we repeated this experiment with the size of the reduced selection set to 10%. The performance of this modified algorithm can be seen in Table 3.1. The heuristic still provided better performance on 837 test graphs, with Infomap and CPM outperforming the greedy method in the same situations as described above. However, the rest of the detection methods only offer the best solution in a few isolated cases.

We can conclude that the overlapping Infomap algorithm dominates the moderately or highly overlapping test scenarios most relevant to our experiment. This dominance is also robust, as it does not depend on the size of the reduced selection set of the heuristic algorithm. Infomap is also a fast algorithm, so the performance overhead introduced by running it on the test networks to obtain the community values is small.

Table 3.1. Number of test graphs (out of 1080) where a community detection method provided the best $\sigma(A_0)$ in conjunction with our heuristic for all overlapping community detection methods. Three variants of the algorithm are shown: the unmodified heuristic with the selection set reduced to 20%, to 10% and the simplified heuristic. We also show the number of times the greedy algorithm of Kempe et al. provided the best result.

	20 % selection set	10 % selection set	Simplified heuristic
Greedy(Kempe)	207	243	335
CPM	71	61	2
COPRA	10	1	0
GCE	0	0	0
Infomap	732	769	737
MOSES	2	0	0
OSLOM	2	2	6
SBM inference	27	0	0
SLPA	29	4	0

3.3.2 Simplified Heuristic Algorithm

Testing for the simplified heuristic was conducted in the exact same manner as in the previous section. Table 3.1 shows the number of test networks, where the heuristic provided a better performance than the original greedy algorithm. We can see, that even if we omit the optimization step, the community-based heuristic still provides better performance on 745 test networks. However, in almost all cases, the Infomap algorithm is the only approach that provides community values good enough to outperform the greedy method. In terms of the properties of these test graphs, we can draw the same conclusions as before. Infomap performs best in moderately or highly overlapping scenarios or when the mixing parameters μ is higher than 0.1

Finally, Figure 3.4 shows the performance of the community detection methods. We can see a notable drop in $\sigma(A_0)$ for all detection method, with the exception of Infomap, which has only a slight drop, further highlighting the robustness of the method. We can conclude that by simply ranking the nodes according to their community values calculated by Infomap and choosing the top k nodes, it is possible to obtain a greater expected fraction of infected nodes than the original greedy algorithm proposed by Kempe et al.

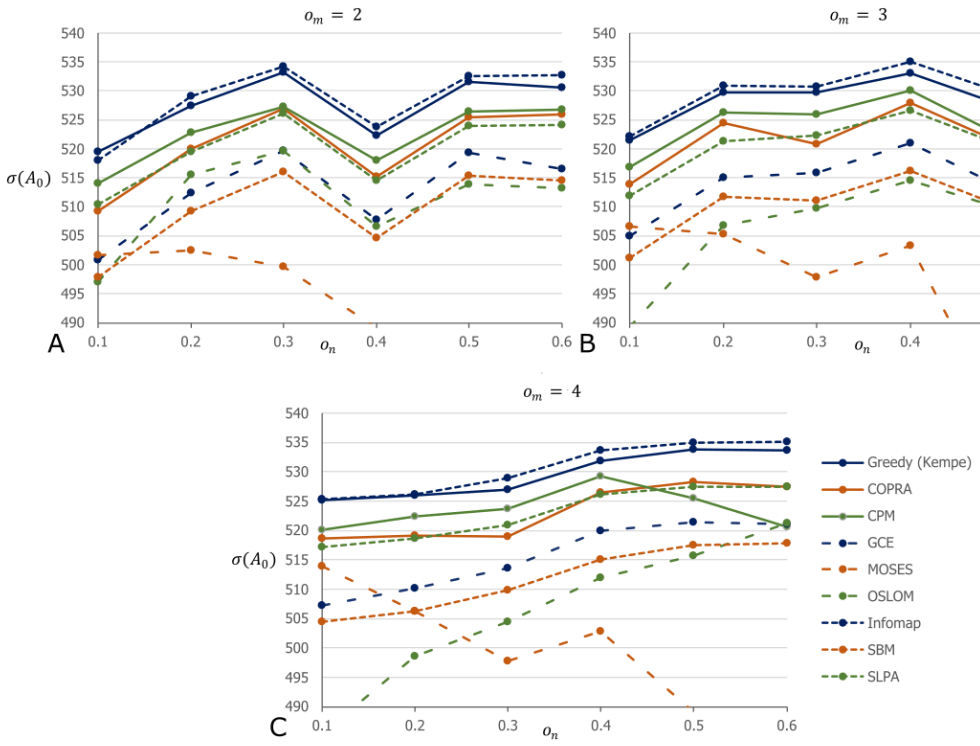


Figure 3.4. Performance of the community detection algorithms in conjunction with the simplified heuristic. $\sigma(A_0)$ is shown for all algorithms with varying values for o_m and o_n, and $\mu = 0.3$. The performance of the greedy algorithm is also shown for comparison.

3.4 Directed Hub Percolation

After we tested the basic idea on different community detection methods, our objective is to examine the methodology on directed networks using two directed community detection methods where we can extract additional structural information about the nodes. First, we introduce the Directed Hub Percolation Method (DHPM) which is an extension of an existing undirected method [28]. We choose this method because it has multiple steps where we can extract additional information about the structure of the given networks in order to improve the infection maximization. Afterwards, a new reduction technique is introduced connected our community detection method. The algorithm is compared with the directed version [141] of the original clique percolation method [140] on generated and real world networks.

The original hub percolation method is based on cliques and hubs. Maximal cliques are maximal fully connected subgraphs of an arbitrary graph, while hubs are locally important nodes in community detection. First, the original algorithm finds undirected maximal cliques containing at least 3 nodes in the network. In our case the clique detection algorithm is replaced by a directed clique

detection algorithm, and an additional parameter is introduced in the end of the method to provide higher resolution results. First of all we define the concept of a directed maximal clique.

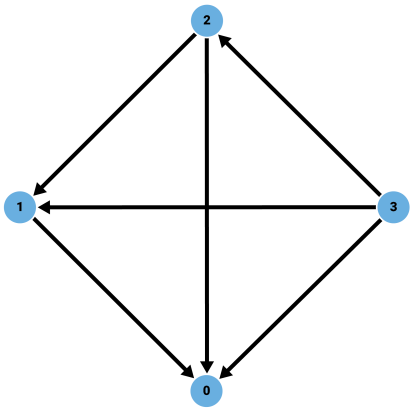


Figure 3.5. An example of a directed clique. The restricted out degree of the nodes are 3, 2, 1 and 0.

Let d_{v_c} be the restricted out-degree of a node v in clique c which means the out-degree of a given node inside the clique. The definition of the directed maximal clique is the following:

- The clique contains all directed edges from v_1 to v_2 where $d_{v1_c} > d_{v2_c}$
- The clique contains no directed loops
- Every node in a clique has a different restricted out-degree
- It is maximal so it can not be expanded to a bigger clique

The Figure 3.5 shows an example of a directed clique. The restricted out degree of the nodes are different from each other. In the literature this structure also called transitive tournament [165][177]. Based on the clique definition, the algorithm of the directed hub percolation method is the following:

1. Find all at least 3 sized maximal cliques in the network. Let C contain these cliques.
2. A HubValue is defined for every node as follows: $\forall v \in V(G)$ let $h_v = |H_v|$ where $H_v = \{h | v \in h, h \in C\}$.
3. Based on h_v and a Hub Selection strategy we decide whether vertices are chosen as hub. Let H be the set of the hubs.
4. Let C_h be the set of the cliques which contains only hubs.
5. Let C_e be the set of extended cliques built in the following way: expand all $c_h \in C_h$ with the cliques containing at least 2 common vertices with c_h, that is with $c \in C$ where $|c_h \cap c| \geq 2$. Let c_e be the subgraph of the expanded vertices.

6. Merge every $c_{e_0}, c_{e_1} \in C_e$ if they have at least x common hubs.

7. The given C_e set contains the communities of the network.

Hub Selection

The third step of the algorithm introduces a Hub Selection strategy, which defines how vertices are chosen as a hubs. The hub selection strategies are the following:

- **Median of 1 neighborhood:** A vertex v is hub, if the value of h_v is greater than the median of the h_v values of its neighbors.
- **Mean of 1 neighborhood with parameter**: A vertex v is hub, if the value of h_v is greater than the average of the h_v values of the neighbors, multiplied by a $q > 0$ parameter.
- **Weighted mean of 1 neighborhood:** The value of the h_v is multiplied by the weights on the out-edges. A vertex v is hub, if the value of the computed h_v is bigger than the average of the h_v values in one neighborhood.

The third strategy was changed compared to the original, emphasizing the direction of the edges, because in a case of a directed network a hub is better from influental point of view if it has more out edges.

3.4.1 Hub Value

Cliques indicate the strongest connection between groups of nodes because in a clique every node is connected to each other. Just as in a case of the community value let $f(v)$ $f : v \to Z$ be a function which assigns a number to each node. Let $f_{hv}(v)$ be a function that assigns the hub value h_v to the nodes of the network indicating how many directed cliques contain the node. The score is based on the idea, that a node can be a good infector if multiple cliques contain it, because in this way the node can spread the infection between cliques. The hub value comes from the second step of the Directed Hub Percolation method.

After every nodes get the score, the $G(V)$ set is sorted according to h_v. The reduced $V*$ set contains the top nodes of the ordered $G(V)$ set. Since the hub value doesn't contain information on the edges of the graph, we introduce two different approaches. Similarly to the community value, the hub value can be used both in weighted or unweighted networks. In weighted case h_v is multiplied by the mean of the probabilities on every out edge of the actual node.

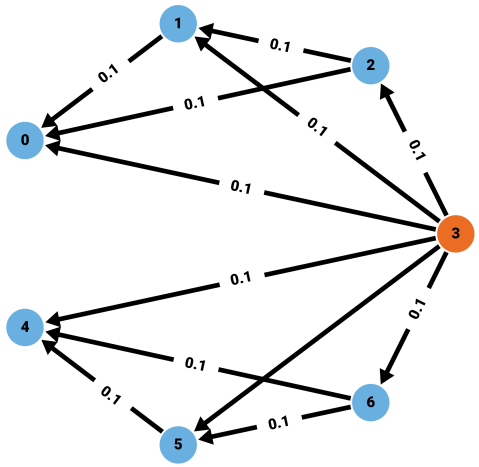


Figure 3.6. Example of hub value calculation. The hub value of node 3 is $f_{hv}(3) = 2$ because two directed cliques contain the red node. All of the green nodes have one as a h_v. In this case the node 3 is a very good infector because it can spread the infection in both directed cliques.

3.4.2 Evaluation of the Directed Hub Percolation Method

In order to evaluate the directed hub percolation method, we use the community values generated by the DHPM and the DCPM as well as the hub value as the byproduct of the DHPM. However, we use the original greedy maximization method again as a baseline in order to evaluate the efficiency of the given scores. The tests were executed on six different randomly generated and seven real-life networks.

Random Networks

The random graphs were generated in six different sizes with the forest fire model [117] using the igraph package in R. The properties of the random networks are the following:

- Number of nodes from 250 to 1500
- Number of edges from 873 to 5205
- Forward burning probability was 0.34
- Edge probabilities were randomly drawn from an uniform distribution between 0 and 0.2

The test results of the original greedy algorithm, and the size of the networks are shown on Table 3.2.

Table 3.2. Results of the original greedy algorithm on random networks

Graph	nodes	edges	k	Greedy	Time
rand_1	250	873	3	20.922	15.81s
rand_2	500	1552	5	37.338	84.31s
rand_3	750	3452	8	93.321	438.52s
rand_4	1000	3708	10	99.462	796.65s
rand_5	1250	4841	13	123.756	1704.85s
rand_6	1500	5205	15	125.044	2534.69s

During testing the size of the initial infected set was 1% of the number of nodes in each scenario. The randomly generated networks are not too big, since our objective in this case to show the usability of our concept and the directed hub percolation method also in directed networks. Furthermore, a real-sized networks often have millions of nodes and edges therefore it is not possible to test the greedy algorithm on them in acceptable time. The size of the reduced set $V*$ was 10% of the number of nodes. Table 3.3 shows the result of the greedy algorithm with the reduced $V*$ based on hub and community based methods. As the size of the networks increases, the running times follow the size of the reduced set. The time column shows that the running times are approximately 10% of the running times of the original greedy method.

Table 3.3. Results for the hub and community based infection maximization algorithm on random networks. (**HV:** Hub Value, **DHP:** Community Value based on Directed Hub Percolation, **DCP**: Community Value based on Directed Clique Percolation, **Diff:** Percentage of the solution compared to the Greedy algorithm, **Time:** Time of the solutions compared to the time of the greedy method)

Graph	HV	Diff	DHP	Diff	DCP	Diff	Time
rand_1	20.87	99.7%	21.03	100.5%	4.65	22%	18%
rand_2	37.35	**100%**	37.67	**100.8%**	9.45	25%	13%
rand_3	93.07	99.7%	93.35	**100.1%**	26.45	28%	11%
rand_4	99.46	**100%**	100.25	**100.5%**	22.63	22%	11%
rand_5	124.06	**100%**	123.1	99.5%	31.15	25%	10%
rand_6	124.9	99.9%	121.4	97%	34.55	28%	10%

In four cases the hub or community value based reduced set method gives a similar or better solution than the original greedy algorithm. However, in the rest of the cases it cannot reach the reference solution but it still gives acceptable results with a much better running time than the simple greedy method. If we compare our two community detection methods, the table shows that the directed hub percolation method gives much better solution than the directed clique percolation. The DHPM detects the overlapping nodes, and the strongly connected dense subgraphs better than the DCPM in these networks. Besides random networks we also tested our methodology on real-life networks.

Real Networks

The first five real-life networks considered in this paper are word association graphs based on a survey connected to the website "Agykapocs.hu" created by László Kovács [102]. The nodes of these graphs are words and the edges are associations between the words based on the user answers. The different networks come from the different versions of the word-association network. The rest of the real-life networks are from a well known data set from Stanford University [119]. The first network from this data set is an email network which describes email connections of a large European research institution [175][118]. The second is the bitcoin alpha trust network which describes trust connections between bitcoin traders [108]. The edge weights of the network were generated in the same way as in the case of our random networks: they were drawn from an uniform distribution between 0 and 0.2.

Table 3.4. Results of the original greedy algorithm on real networks

Graph	nodes	edges	k	Greedy	Time
real_1	2751	5043	28	215.955	8969.39s
real_2	2088	3839	21	141.533	3868.12s
real_3	1680	3082	17	95.563	2005.33s
real_4	1460	2632	15	75.568	1298.91s
real_5	1305	2315	13	61.137	889.64s
email	1005	25571	10	670.187	5742.73s
bitcoin	3783	24186	38	936.535	83303,53s

We can find lot of real-life networks larger than these, but according to Table 3.4 the running times can be very high even on these relatively small networks, indicating that the normal greedy algorithm may not be able to find a good solution especially with a higher k parameter. The results are shown in Table 3.5.

Table 3.5. Results for the hub and community based reduced set algorithm on real-life networks.(**HV:** Hub Value, **DHP:** Community Value based on Directed Hub Percolation, **DCP**: Community Value based on Directed Clique Percolation, **Diff:** Percentage of the solution compared to the Greedy algorithm, **Time:** Time of the solutions compared to the time of the greedy method)

Graph	HV	Diff	DHP	Diff	DCP	Diff	Time
real_1	215.05	99%	215.8	**100%**	189.6	87%	10%
real_2	139.92	99%	138.6	98%	122.8	87%	10%
real_3	99.46	**104%**	97.78	**102%**	87.69	91%	10%
real_4	74.87	99%	74.34	98%	71.64	94%	10%
real_5	61.18	**100%**	59.54	97%	58.54	95%	11%
email	662.5	99%	662.4	99%	663.4	99%	10%
bitcoin	924.7	98%	916.4	98%	928.1	99%	11%

On real-life networks the results are not as pronounced as in the case of random networks. In three cases our method reaches better solution than the greedy method, and in the remaining four

cases the obtained values are slightly under the results or the original greedy. It can be seen that the Directed Clique Percolation method performs much better on real world networks than on our randomly generated networks. As it is expected the running time is again around 10% compared to the running time of the original greedy method.

3.5 Summary

In this section we examined the properties of a community-based heuristic for influence maximization, and proposed a simplified variation with a greatly reduced running time. In the first part of the section, we have used the output of eight well-known overlapping community detection methods on a large variety of undirected test networks with different community structures. In the second part, we tested and extended our methodology on directed networks, introducing a directed version of an undirected community detection method. We compared the expected size of infected nodes obtainable with our heuristic with the original greedy method of Kempe et al. Our approach can also be used to measure the applicability of overlapping community detection methods for the task of influence maximization.

According to our results, both the original and the simplified heuristic is able to provide a greater expected size of infected nodes than the greedy algorithm. The simplified heuristic completely omits the optimization step of the greedy algorithm, reducing its runtime considerably. Our results also indicate, that the Infomap algorithm provides the best performance for this task by a large margin. This detection method excels in networks with a moderate or large amount of overlaps, and its performance stays robust even when used in conjunction with the simplified heuristic. Our methodology works both undirected and directed networks, and the basic idea that scores the nodes based on their role in the different communities proves to be effective in a case of infection maximization. We also used the discussed methodologies in the following research papers [104, 79, 103]

Chapter 4

Uplift Network Model for Targeted Interventions

Diffusion and infection models on networks are suitable to express not just the spread of viruses or information, but also the behaviour of real-world processes between given entities. As we showed in the previous section, maximizing the spread of the infection or influence can be useful if the objective is to disseminate new information on the network. However, the objective often can be the minimization of existing negative network processes through interventions. Since recent global events highlighted the vulnerability of the economy, companies and the health system in a case of a negative and overstrained global situation, we think that the interventional strategies can be crucial in negative or recessional situations. In order to understand the different scenarios and the effect of different props or interventions policy makers can use to minimize negative effects; the key is to understand the processes and relations between our entities. Optimizing these strategies can bring many advantages on the network level and provide efficient decision support system for the policy makers, governements and company owners. Our objective is to give a general model that is able to support decision making processes, where the goal is to minimize different network effects using intervention strategies.

In this chapter, we introduce a new Uplift Network Model through a psychological use case where the objective is to improve the mental wellbeing in organizations and companies with targeted psychological interventions. Therefore the network effect that we want to minimize is the negative well-being, while the nature of the intervention is truly psychological. First, we will introduce the psychological background of the problem and the challenges that inspired our model. In the second part of the chapter our model will be introduced and defined while at the end, the results of the method will be introduced using real data from 14 Norwegian nursing homes. Regarding to the Uplift Network Model it is important to note, that the model can be used in many different areas,

such as epidemiology, where vaccination strategies can be optimized and economy, where churn or negative recessional effects and bankruptcies can be minimized in the network.

4.1 Psychological Background

Mental health of employees is critical for the long-term success of an organization. Poor employee wellbeing can lead to undesirable outcomes, including absenteeism, loss of productivity and increased health insurance costs [41]. It is not surprising that various attempts have been made to improve (mental) health in organizations. These approaches can be grouped in organisational-level and individual-level interventions. The former group strives to improve physical environment (e.g., decrease noise), work time conditions (e.g., pace of work), and organisation conditions (e.g., structure of hierarchy) [131]. The latter group aims to equip individuals with knowledge and skills to better cope with work conditions (e.g., stress management classes) [110].

While both approaches are valuable, they each bring their own set of obstacles. Organizational-level interventions are advantageous in simultaneously addressing the entire group of employees, but they often have little or no effect [131, 22]. On the other hand, interventions aimed at individuals (particularly cognitive-behavioural programmes), can reliably lead to significant positive changes but are less efficient, as they often need to be administered over several weeks in either small groups or one-on-one [152, 168]. Despite their effectiveness, sizeable costs required to provide such interventions to all employees might discourage organizations or companies in offering them.

In such cases, a potentially valuable option is to offer only a limited number of individual-level interventions, but in a way that could benefit even individuals that themselves do not receive an intervention. This could be achieved by targeting individuals selected based on their ability to "infect" mental wellbeing of other individuals with their own. The approach thus suggests exploiting the phenomenon of mental health contagion – the observation that mental health of a particular individual can influence other individuals [51]. Considering this, a psychosocial intervention could not only improve wellbeing of a highly contagious individual but also positively affect surrounding persons. The mechanisms behind the contagion of mental health are numerous and in complex interaction, among them are social comparisons, collaborative development of negative interpretations of recent events and spreading of (unpleasant) affective states (i.e., core affect, emotions and mood) [51]. As an example, consider how affective states can be involved in the contagion process. Unpleasant emotions, such as anger, fear or sadness, can be "transmitted" between individuals, because people tend to unconsciously mimic facial expressions, voices, movements and behaviours that can all influence affective states [83]. Chronic experience of such unpleasant affective states (and the lack of pleasant emotions), could contribute to developing mental disorders [61, 62, 63].

The effects of contagion can expand beyond influencing the wellbeing of other group members and can influence group dynamics as a whole, including the attitudes and behaviours of work teams.

It has been shown, for example, that when a trained confederate successfully "infected" experiment participants with pleasant affective states, the cooperation between team members increased and conflict decreased [17]. Clearly, transitory affective states seem likelier to spread between individuals than more stable and enduring states of mental health. Yet it is important to keep in mind that prolonged subtle effects (e.g., increased sadness) could add up to a substantial overall effect (e.g., symptoms of depression) [61]. Indeed, several studies have observed that the overall mental health of an individual is influenced by the mental health of surrounding people [51, 60, 154]. Current findings imply that the contagion of mental health could be utilized to increase wellbeing of a larger group of people by targeting only few selected individuals. Considerably improving mental wellbeing of the most contagious people could be a more cost-effective approach to improve overall wellbeing of the entire personnel, when compared to directly but slightly improving wellbeing in each employee (as could be achieved with certain organisational-level interventions). The contagiousness of individuals depends on many factors, both personal and contextual. Important personal characteristics include contagion ability (e.g., emotional expressiveness) and susceptibility [39]. Contextual factors include, for instance, the nature and amount of time individuals spend together. Supervisors are an obvious example of individuals who might be especially prone to being contagious, as they tend to be important in the lives of their subordinates and ordinarily have many social connections [40, 51].

Our objective is to examine if, at least in theory, selectively targeting highly-contagious individuals with psychosocial interventions can be an efficient solution to improve overall wellbeing of a larger group of people. We will explore this by running social network infection simulations based on empirically derived effect sizes representing real-life effects of psychosocial interventions and the degree of mental health contagion. The simulation of the infection process can show if the overall wellbeing of the entire group is disproportionately improved when highly contagious people are targeted with psychosocial interventions (compared to randomly selected individuals).

4.2 Uplift Network Model

To define the Uplift Network Model formally, let $G = (V, E)$ be an undirected or directed network, where $\forall(v, u) \in E$ edge has a $p(v, u)$ probability where $0 < p(v, u) \leq 1$ and $\forall v \in V$ node has an a priori $v^{apriori}$ and an uplift v^{uplift} probability where both of them are between 0 and 1 so $0 \leq v^{apriori} \leq 1$ and $0 \leq v^{uplift} \leq 1$. The basic idea of the model is an extension of the Generalized Cascade Model proposed by Bóta et. al. [29] where nodes can become infected randomly in the beginning of the process. The spreading process starts with the A_0 initially infected set similarly to the original Independent Cascade Model, however, the nodes can be in A_0 based on their $v^{apriori}$ probabilities. The pseudocode of the Generalized Independent Cascade can be seen in Algorithm 7.

The pseudocode shows that the iterative infection process is the same as in the case of the original Independent Cascade algorithm. Let $\sigma(V)$ be the expected value of the infection process. Since the

Algorithm 7 Generalized Independent Cascade

1: $A_0 \leftarrow \emptyset$ be the set of initially infected nodes
2: **For** v in $G(V)$
3: $\quad A_0 \leftarrow v$ based on $v^{apriori}$
4: **End For**
5: **While** $A_i \neq \emptyset$
6: $\quad A_i \leftarrow$ newly infected nodes
7: $\quad \forall v \in A_i$ tries to infect their neighbors with $p(v, u)$
8: $\quad$ **If** the infection is successful
9: $\quad\quad A_{i+1} = A_{i+1} \cup u$
10: $\quad$ **End If**
11: **End While**

exact computation of the infection value is #P problem [37] and with simulation any precision can be reached, we again use a simulation to computer the $\sigma(V)$. The pseudocode of the modified simulation is defined in Algorithm 8.

Algorithm 8 Generalized Complete Simulation

1: **Input:** Graph G, sample size s
2: $A_0 \leftarrow \emptyset$ be the set of initially infected nodes
3: **For** v in $G(V)$
4: $\quad A_0 \leftarrow v$ based on $v^{apriori}$
5: **End For**
6: $j \leftarrow 0$
7: $\forall v \in G(V) : f_v = 0$
8: **While** $j < s$
9: $\quad \forall e \in G(E)$ let the edge active or passive based on $p(e)$
10: $\quad$ Modified DFS from $\forall v \in A_0$
11: $\quad$ **If** $n \in G(V)$ node is accessible from $v \in A_0$
12: $\quad\quad n : f_v \leftarrow f_v + 1$
13: $\quad$ **End If**
14: $j \leftarrow j + 1$
15: **End While**
16: $\forall v \in G(V) : fv \leftarrow \frac{fv}{s}$

The lines 2-5 is corresponding to the selection of the initial infected nodes. It is easy to see that if the $v^{apriori} = 0, \forall v \in V$ then the $\sigma(V) = |V|$ so each node will be infected, while if $v^{priori} = 0. \forall v \in V$ then $\sigma(V) = 0$ so there is no infection in the network. In the following section we define the Intervention Optimization problem on the previously defined network.

4.2.1 Intervention Optimization

The Intervention Optimization problem represents a real-word scenario, where a diffusion process is already spreading on the network and our objective is to minimize the effect of the process using

limited amount of interventions. The $v^{apriori}$ probabilities are used when the node did not get intervention, while the v^{uplift} means the probability of the node after intervention. To define the general model, we separate the $G(V)$ into two subsets along the intervened nodes.

- $G(V, E)$ previously defined network
- $I_{apriori} \subset G(V)$ node set without intervention
- $I_{uplift} \subset G(V)$ node set with intervention
- k number of possible interventions, so the expected size of I_{uplift} is k.

If the $I_{uplift} = \emptyset$ and the $I_{apriori} = G(V)$ so the apriori set contains all of the nodes from the network, then it means that there is no intervention. The expected value $\sigma(I_{apriori})$ defines the outcome of the infection without any intervention and gives us a reference expected objective value that we want to minimize during the optimization process. If the optimization environment can be seen in Figure 4.1.

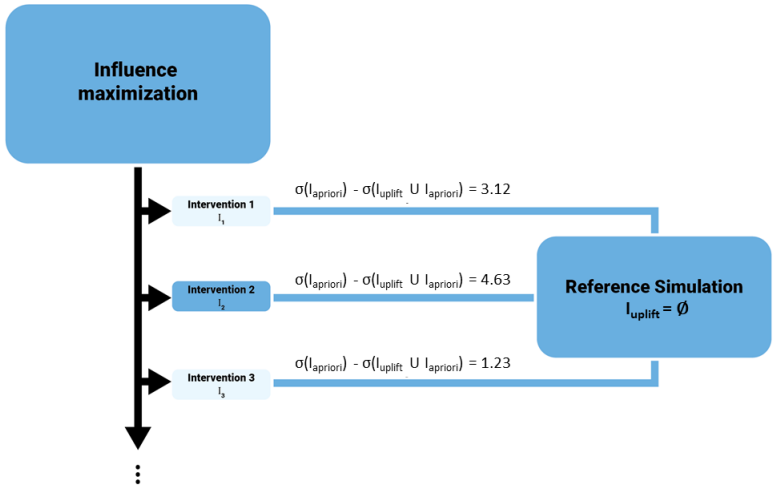


Figure 4.1. Optimization environment

If the node is in the $I_{apriori}$ set, the simulation uses the $v^{apriori}$ probability of the node, while if it is in the I_{uplift} set, the v^{uplift} value of the node is used. To evaluate the intervention, the environment compares the actual solution to the reference simulation where $I_{uplift} = \emptyset$ and we maximize the value of the $\sigma(I_{apriori}) - \sigma(I_{uplift} \cup I_{apriori})$ formula where the first part is the reference simulation while the second part is the intervention with the nodes in the I_{uplift} set. The example on the figure 4.1 shows the objective value of three interventions where the value means that the different interventions

decreased the value of the infection by 3.12, 4.63 and 1.23. It can be seen that the second intervention gives us the best solution in this case, and maximizing the given formula minimizes the infection on the network. During the optimization process we used the greedy method again [95] with a modified objective function. The pseudocode of the method can be seen in Algorithm 9.

Algorithm 9 Greedy method

1: **Input:** Graph $G(V, E)$, size of the intervention set k
2: **Output:** I_{uplift} intervention set
3: $I_{uplift} \leftarrow \emptyset$
4: **While** $|I_{uplift}| \leq k$
5: $\quad I_{uplift} = I_{uplift} \cup \ arg\ max_{v \in I_{apriori} \setminus I_{uplift})}(\sigma(I_{apriori}) - \sigma((I_{uplift} \cup \{v\}) \cup I_{apriori}))$

The algorithm increases the size of the intervention set in each iteration, while it reaches the expected size. The greedy algorithm chooses the node for the intervention that maximizes the decrease of the infection in the network. In the following chapter we introduce the use of our method through a psychological use case.

4.3 Psychological Use Case

As we discussed before, the proposed methodology might be valuable in optimizing wellbeing of large groups, by exploring how the mental wellbeing of all people in the group changes in response to providing psychosocial interventions to different individuals. It is important to point out that the extent of potential changes in wellbeing relies heavily on the parameters of the model (e.g., degree of mental contagion), which could substantially differ between social contexts. Thus, although the approach will be presented through empirical data, we would like to emphasize that the method might be useful and worth exploring in other contexts with different parameters, such as different characteristics of individuals (e.g., age, gender, personality) and the environment (e.g., proximity of other people). In this section, we introduce the psychological use case, where we use real world data to create a social network and aim to improve the overall wellbeing of the entire group of employees by providing a psychosocial intervention to the most contagious people that had been selected based on several parameters. The overall wellbeing score resulting from the simulation was compared with the score from the simulation in which individuals recieve the psychosocial intervention randomly. Each step is presented in more detail below.

4.3.1 Data Collection and Transformation

We collected the data on 414 employees from 14 nursing homes in Norway, who completed the survey capturing demographic data, work-related information (e.g., occupation, working years, working hours, shift work) and levels of wellbeing.

We used only the data from participants that had completed the questionnaire assessing wellbeing and who, based on the assumptions of our network model, had at least one social connection. These conditions were met by 278 people (268 women), with the mean age of 46.94 years (from 19 to 70; SD = 11.375). Most persons were employed as nurses and auxiliary nurses (235), followed by other healthcare workers (24), supporting staff (14), and managers (3).

Wellbeing was assessed with the WHO-5 questionnaire [172] that asks five questions pertaining to the subject's last two weeks (e.g., "I have felt cheerful and in good spirits."). Subjects answered each question on a six-point Likert-type scale (0 = "At no time", 5 = "All of the time"). The results for one item of the questionnaire ("I have felt calm and relaxed.") were missing in our data, so we calculated the final score from the remaining four items. We summed the values of responses to obtain the raw score and then rescaled it to obtain a percentage score ranging between 0 and 100 (larger number represents higher levels of wellbeing). In our sample, the mean percentage score on WHO-5 was 68.09 (SD = 17.17). For the purposes of the simulation, however, the score was first divided by 100 to obtain values between 0 and 1 and then reversed, so the values closer to 0 represent higher levels of wellbeing. This reversal was necessary due to the nature of our uplift network model.

To give an instance of a general model, we used the collected data from the nursing homes. Since we did not have information about the real connections between the employees, we created the connection structure based on similarities of different individuals. In the network, every employee is represented by a node and the connections between them were assigned if:

- They were employed at the same nursing home
- They had the same occupation (e.g., nurse)
- The age difference between them was not greater than 20 years.

The strength of the connection was computed based on properties of the corresponding employees, resulting from the sum of the following properties:

- Age difference: Difference in age of the corresponding employees, where lower age difference increases the connection strength, scaled between 0 and 0.33
- Matching work shifts: The probability of employees meeting during work due to similar work schedules, where matching shifts increase the connection strength, scaled between 0 and 0.33
- Weekly working hours difference: The probability of employees meeting during work due to similar working hours, where a similar number of working hours increases the connection strength, scaled between 0 and 0.33

The final edge weight is the sum of the scaled values, so a number between 0 and 1, multiplied by a random number between 0 and 0.021, which represents the extent of mental health contagion.

The resulting network had 289 nodes and 731 edges. A sample of the network is presented in Figure 4.2. Nodes, representing employees, are coloured based on the rescaled and reversed WHO-5 scores; the width of the edges increases with the edge weight (i.e., probability of the mental health contagion).

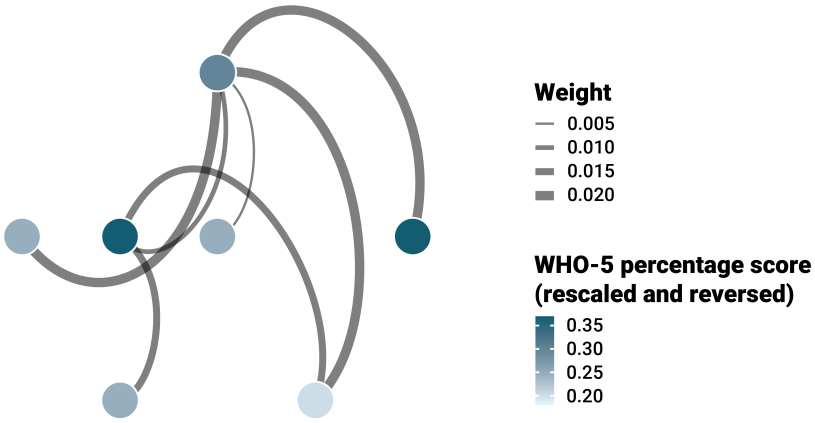


Figure 4.2. An example of the created network

4.3.2 Effect Sizes Used in the Simulation

Effect of the Intervention

To understand the used node probabilities so v^{uplift} and v^{uplift} it is important to define the effect size of an intervention in psychology so the average effect of the intervention on an individual. The meta-analysis of various individual-level interventions in organizations reported that cognitive-behavioural programmes produced the largest average effect size (Cohen's d = 1.1154) for a combined group of mental health outcomes that included measures of stress, anxiety, mental health, and work-related outcomes (e.g., work satisfaction, motivation, perceived control) [152]. This effect size was incorporated in our model; on average, every employee targeted by an intervention had their wellbeing score increased by 1.1554 multiplied by the standard deviation of the WHO-5 percentage score rescaled between 0 and 1 (in our case, SD = 0.172). To approximate varying effects expected in real-life, the effect size of the intervention provided to each employee varied according to the normal distribution with the mean of 1.1554 and standard deviation set at 0.10.

Mental Health Contagion Effect Size

Among the identified studies examining mental health contagion [51, 60, 154], we selected a study from Eisenberg et al. [51] that was especially careful in controlling several sources of bias and,

correspondingly, arrived at a lower estimate of the mental contagion effects compared to other studies ($\beta = 0.053, SE = 0.031$). Although this effect size is based on specific anxiety items from the K-6 instrument assessing general psychological distress [97], the captured construct has been shown to have a considerable overlap with the construct tapped by WHO-5 [48]. For our simulation, the selected effect size indicates that the wellbeing score of a neighbour in a social network will increase for 0.053 multiplied by the standard deviation of the WHO-5 percentage score rescaled between 0 and 1 (SD = 0.172). To allow for varying degrees of contagion based on the strength of social connections (e.g., amount of time spent together), we instructed our simulation model to select a value from the 95% confidence interval [0, 0.12] of the included effect size, where the stronger social connection received a higher value. The resulting values that are used in the simulation thus lie on the interval between 0 (i.e., 0.172 * 0) and 0.021 (i.e., 0.172 * 0.12).

4.3.3 Results

To analyse the effects of providing interventions selectively to highly contagious individuals, we compare that approach with administering interventions to randomly selected individuals. Figure 4.3 displays the mean increase in WHO-5 percentage score per person after the hypothetical intervention was provided (compared to the reference point scores without the intervention). The figure separates the scores based on the order in which the interventions were administered; in one case, the order of individuals provided with an intervention was random, in the other, the interventions were first administered based on our optimization environment. In both cases, the scores steadily increase until all individuals receive the intervention, where the average increase becomes similar to the effect size of the intervention used in the simulation. When the intervention was administered to highly contagious people first, the increases in scores were generally larger when compared to the scores following random administration of interventions. This represents the effect of contagion: although, on average, the score of each individual increased (i.e., improved) the same after the intervention, some persons were better at spreading that improvement to others, due to their contagiousness.

Some of the results from Figure 4.3 are presented in more detail in Table 4.1. The table displays the exact mean increase in WHO-5 percentage score for selected numbers of provided interventions. After 20 interventions, for example, the percentage score increased, on average, by 1.61, when intervention administrations were ordered by contagiousness, which is 0.18 larger than the increase following randomly administered interventions. In this case, selectively targeting contagious individuals is thus responsible for an 0.18 increase WHO-5 percentage score, all else being equal.

The observed differences in mean scores between random and targeted intervention administrations are small. This is not surprising, given that our model was based on a relatively large effect size following an intervention, but only a fraction of that improvement was expected to be transmitted between individuals, due to the small effect size of mental health contagion that was incorporated in the Uplift Network Model. However, in contrast with the intervention effect size that was based on a

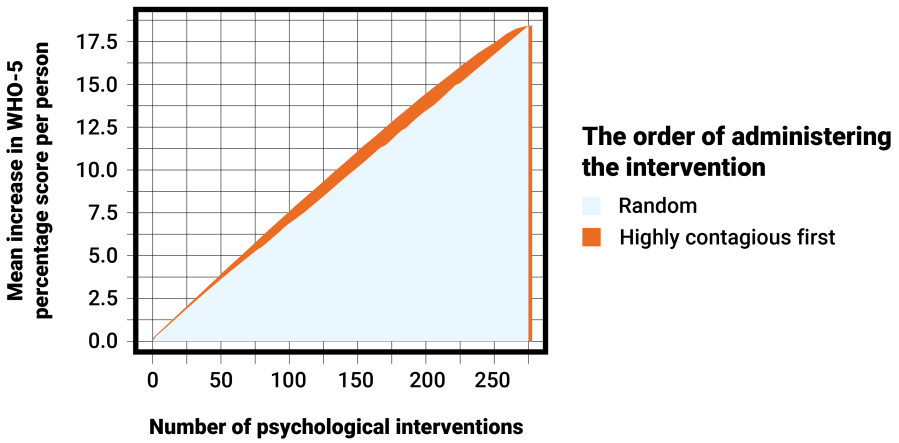


Figure 4.3. Mean increase in WHO-5 depending on the order of intervention administration

Table 4.1. Comparison of the WHO-5 percentage score mean increase per person

Number and percent of interventions	Mean increase in score after random administrations	Mean increase in score after targeted administrations	Difference between targeted and random
10 (3.6%)	0.74	0.83	0.09
20 (7.2%)	1.43	1.61	0.18
50 (18.0%)	3.55	3.94	0.38
100 (36.0%)	6.85	7.64	0.79
200 (71.9%)	13.58	14.64	1.06

meta-analysis considering several studies, the contagion effect size was based on a single study [51], due to lack of relevant research. Despite the robustness of that study, there are reasons to assume that the contagion effect could be larger. As is generally the case, a single result can rarely be a definitive answer on the topic. Indeed, other studies researching mental health contagion in other contexts have arrived at considerably larger effect sizes (although, admittedly, in those studies the potential for bias was higher) [60, 154]. Another important aspect is the social context. Our model used an effect size derived from a study examining contagion in college roommates. In different contexts, however, the contagion effect could be larger, as it may depend on various individual and interpersonal factors [40]. Difference in social status is one of the factors increasing the degree of contagion; contagion is more likely pronounced when passing between a higher social status individual and one with a lower status (e.g., between a supervisor and a subordinate) [40]. Presumably, such asymmetries in social status are more common in many hierarchically structured organizations than in relationships between college roommates, on which our contagion effect size was based. If the contagion of mental wellbeing is indeed more pronounced in certain organizations, our Uplift

Network Model could show that the overall effects of contagion are considerably larger compared to the effects reported in this article. The effects could be particularly pronounced, for example, in highly hierarchical organizations, where supervisors hold especially high status compared to their subordinates (presumably leading to larger contagion effects) while at the same time supervising many employees (i.e., there are numerous recipients of the contagion effects). An organization could identify such potentially highly contagious individuals with an Uplift Network Model, assuming appropriate data is available or can be collected. An example of such identification is presented in Table 4.2. The table displays the basic information on the top three most contagious person in our sample, those who had the highest positive impact on the wellbeing of the entire group after receiving the intervention. If only a limited number of intervention is provided, intervention targeted at such individuals can lead to the largest improvement in the wellbeing of the entire group.

Table 4.2. Basic informations about the most contagious employees

Gender	Age	Education	Occupation	Weekly working hours	Years employed
Male	41-50	University or more	Nurse specialist	31-40	0-5
Female	61-70	University or more	Nurse specialist	31-40	0-5
Female	51-60	Highschool or less	Auxiliary nurse	31-40	21-25

4.4 Summary

Decreasing negative network effects in a system and organizing interventions effectively can be a challenging task. In this chapter we gave a new infection based model, the Uplift Network Model, that is able to optimize the targeted intervention strategies in a system where the connection structure of the entities can be depicted with a network. Whereas the use case where the motivation comes from the field of psychology. Improving human mental health is a challenging task, especially when attempting to improve wellbeing of a large group of people. Often interventions can lead to only minor improvement in regard to the overall wellbeing in the workplace. Organizational-level interventions can address the entire personnel simultaneously, but provide little effect, while interventions targeting individuals provide considerably larger effects, but can require substantial resources in terms of time, effort, and money. Either way, regardless of which intervention type is selected, many individuals will be in need of additional support. Clearly, in this case, the efficiency of interventions is of interest. One way to increase the efficiency of existing interventions is administering them to specific individuals – those who are highly contagious and can 'transmit' their mental wellbeing to other people. In effect, those individuals can make the most of the intervention, as far as the overall wellbeing of a group is concerned.

We have seen, however, that the effects of the mental health contagion can be relatively small and that singling out individuals, who are selected to receive the intervention, might bring additional challenges. Yet it is important to keep in mind that in different contexts the contagion effects could be larger and that issues stemming from singling out individuals might be well worth the price, considering the subsequent improvement in overall wellbeing of the group. We have shown that, at least in principle, the wellbeing of a group can be more efficiently improved if highly contagious people are targeted with interventions. As this approach could improve the efficiency of psychosocial interventions, leading to improved wellbeing in organizations, it is worth further theoretical and empirical exploration.

Chapter 5

Temporal Network Analytics for Fraud Detection in the Banking Sector

In the banking sector the financial institutions developed and digitized their infrastructure and services over the decades. Presently a dynamically increasing number of tools are available for the customers to improve the service level. From the clients point of view, this evolution offers a comfortable and safe control over their financial tasks. However, as a by-product, through the established system, nowadays financial institutions are able to collect data about our financial activities. The produced data of the client activities contain important information about the use of the actual financial system, and can be useful in many different ways [25, 106]. With the help of the extracted information, the institutions can offer personalized products, improve the quality and traceability of the whole system, maximize their profit or even detect and prevent the illegal fraud activities of individuals or companies. From the the financial institutions point of view, the key challenge is the extraction of the suitable and required information from the activity data.

In this chapter of the dissertation, we define a special case of the banking fraud, the fraud through cycle in the transaction network and introduce a method which is able to detect the suspected fraud activity in this network. Different fraud techniques can be connected with the cycle transactions such as if a group of companies makes financial transactions along a cycle through false orders, and for each transaction they book false expenses to avoid the taxation and "launder the money". The other typical fraud with this method is, if a company or an individual needs to prove that having enough fund or spare to get a loan. Nevertheless, in the banking sector there are many other versions of the frauds through the cycle transactions.

The main motivation behind this research was a real world request from an anonymous bank which integrated a fraud module in its information system and as a key element of this module it was requested to detect transaction cycles to identify strange usage of the system. The developed methodology integrates the temporal network topology analysis with the relevant connected data, which is the amount of the transfer in this case. The above characteristics of this approach can set the basis of new solutions for future use of temporal network queries in graph databases. The research methodology follows the guidelines for conducting and evaluating design-science research [86, 144].
The structure of this chapter is the following. After the Introduction, first we give an overview of related work, then we will present the concept of fraud detection with cycle search. In Section 5.3 the main contribution is described by giving the formal problem definition and the detailed summary of the methodology. In Section 5.4 the use case is presented on real data provided by a Hungarian bank. Finally we close the chapter with a summary.

5.1 Short Overview of Fraud Detection

Data analytics based fraud detection solutions on digital data has a relatively long history. Banks together with telecommunication and insurance companies were among the first in the use of statistical based methods in the industry [19]. However, it turned out that even with significant amount of data (which was already available decades ago in the above sectors) the statistical inference with fraudulent activities causes one of the main challenges: there is a high number of legitimate records for each fraudulent one. Therefore AI based anomaly detection methods have become a central approach in the field [171].
Business information systems containing fraud detection module need to consider different layers of customer actions. According to Gartner [121] five layers can be distinguished (see Fig 5.1). As it can be seen, for the detailed analysis an in-depth network structure of the problem (Layer 5) is required. This structure reflects the consequence of the above mentioned statistical inference: using individual data analysis has a limitation for identifying fraudulent patterns.
Motivated by these constraints and the explosion of network analysis in the last 20 years, recently a few graph based solutions have been developed for fraud detection (see e.g. [142, 33, 148]). However, these solutions were applied on online networks of data streams for identifying new simple patterns, and did not consider the temporal dimension.

The significance of network based methods is also reflected by the evolving development of graph databases, which need sophisticated methods for structural graph queries [20]. The intensive business use of these platforms naturally raises the question with respect to their applicability for fraud analysis [157]. However, standard solutions are not available for two reasons. On one hand, the network based methodology is not widely adapted yet; on the other hand, the time dimension has

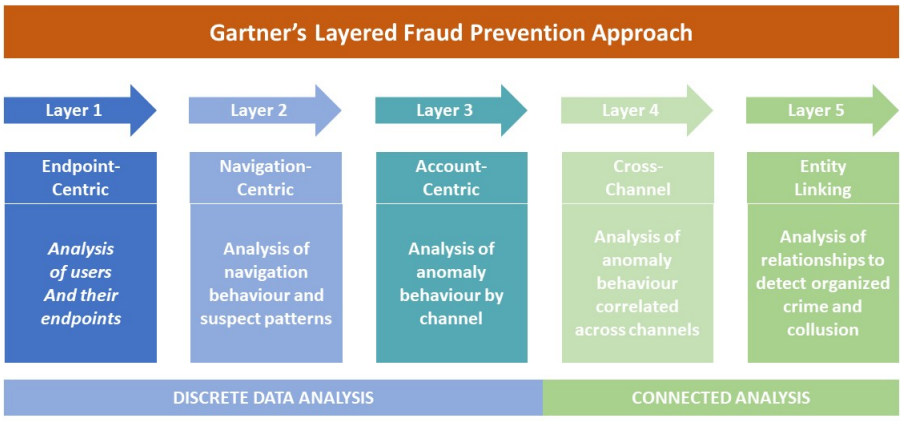


Figure 5.1. The 5 layers of fraud detection, based on Gartner [121]

a central issue in fraud detection, which is handled on data level only, but not with an integrated manner in network topology. Apart from a preliminary case study [31], no specific approach has arisen in this direction.

The basic barrier for the technological development discussed in the previous paragraph is the sporadic results on efficient algorithmic methods towards graph structure queries in temporal networks. Nevertheless, in recent years a few efficient algorithmic solutions were developed for networks where edge labels have time stamps. The authors of [143] worked out a general methodology for identifying different graph motifs. However, this approach is too general for specific questions like cycle detection. The approach of [107] is targeting related specific problems, but their method is identifying simple cycles (no repetition in nodes) only, and the related data (attributes of the nodes and edges of the network) are not considered in the potential queries.

Concerning the classical methods for cycle detection and elementary cycle (simple cycle without internal node repetition) detection in static networks, two different approaches were investigated. The general cycle detection [167] is mostly based on a depth-first-search, and a cycle is found from a given node if the DFS [166] finds a back edge to the ancestor node. The main problem with this approach is, it concentrates on the nodes and not on the edges, but from our point of view the edges contain the important information about the financial transactions. The other approach [92] is the elementary cycle detection, which lists all of the cycles in the network. The above methodologies are also extended for the analysis of large graphs in a distributed manner [153]. Even though these approaches principally could be used for the fraud detection, because in the elementary cycle problem the main emphasis is on the edges, but they cannot be applied directly as in our problem the

timestamps and the specific attributes (e.g. transferred amounts) are also important.
In summary, relevant results were published on AI based fraud detection, graph databases, temporal network pattern recognition and static graph cycle detection, but an integrated approach is not available yet.

5.2 Network Based Fraud Analytics in the Financial Sector

In most cases the financial institutions have the proper data about the client activities and financial transactions, but they are not able to extract the information needed to identify the suspected fraudsters. In ”money laundering”, the fraudsters try to hide the illegal source of their income in order to transform into legal one. Financial institutions are interested in ”fighting against” this situation, because the reputation is highly important in this area: if a fraud is revealed publicly, it can cause a huge loss indirectly as well. The inflow of illegal financial source into the legal economy can distort the balance of financial markets, thus identifying frauds is a common interest of the banking sector. In the real life the data mining based fraud detection methods are very important tools to identify the fraudsters based on the produced data, thus extracting and detecting the suspected patterns in the financial system is a key task. A good overview about the financial frauds and a classification of the methods can be found in [137] and [171]. It can be stated that most of the financial fraud detection systems are based on machine learning methods, which can be powerful tools to classify the clients into different groups with respect to ”fraud-like” activities. Our detection method can be part of this type of systems and can explore the frequency of clients in transaction cycles.
From the government point of view, detecting frauds in the banking system is also very important: most of the tax evasions and fictitious companies are detectable through the banking data. A fictitious company is a unreal company which can be identifiable if it has the following properties. The headquarter is in an address where hundreds or thousands of companies are registered, the owner has unrealistically many companies, and the company is in connection with transaction cycles where tax evasions and cycle billings can happen. In our research the transaction cycles are also important from this point of view. These types of frauds are often in connection with transfer cycles; the detection of the cycles can support the institutions and with the extracted information it is possible the detect the fictitious companies.
The transactions are executed in a "continuous manner" with respect to time dimension and not every transfer is important for the bank. Since the frauds happen in a way that through the detected cycle the amounts on the edges can differ from each other, but the amounts are close to each other, so there is a gap between transfers, it possible that a cycle is a valid suspicious activity from one node, but invalid cycle from another. In summary, it can be stated that the fight against the fraud in the banking sector is really important and can be a big challenge. Therefore it is very important to

develop new methodologies to improve the efficiency of detection systems and identify the suspected fraudsters in an integrated manner of temporal network topology and data.

5.3 Problem Formulation and Solution Methodology

It can be seen by the previous section that monitoring fraudulent activities in financial networks of banking systems is a crucial business question in everyday operation. We have shown that cycle detection in temporal network data can play a key role in this process. In this section we will formalize the investigated problem and will provide the detailed description of our new solution methodology.

5.3.1 Financial Transaction Network

The financial transaction network has two different aspects. The first one is the transaction itself in which there is a transaction data, where each transfer is recorded. It means that for every transfer we have a "from" and "to" account, an amount, and a date when the transfer was realized. As the other aspect we have a data table about the clients which contains key-value pairs, where the key is the client and the values are the corresponding bank accounts to the actual client. One bank account can belong to one client (individual or company) and one client can have multiple accounts. First let B be the set of the bank accounts, where each $b \in B$ is an account identified by the IBAN number. To define the first network let $G(B, E)$ be a directed network where B is the set of the bank accounts, so every node is an account in the bank and E is the set of every transaction among the bank accounts. The transactions have different attributes, consequently let t_e denote the timestamp and a_e be the amount of the edge e, where $e \in E$ and e is a transaction which realized in time t_e and the transferred amount is a_e through the transaction. An example of the network G is shown on Figure 5.2. a).

Since a client can have multiple account in the bank, and in the fraud detection the cycles are interesting between the clients and not just between the bank accounts, let C be the set of the clients. Lets define a new network which is a directed hypergraph $H = (P, E)$ where every $p_i \in P$ is a set for all $i \in C$ so p_i is the set of the client i, so it contains every bank account which corresponds to the actual client, and E is the set of the edges from the network G. The input of our algorithm is the hypergraph H, so the nodes are the clients where every node contains each bank account of the actual client, and the edges are the original transactions between the accounts.

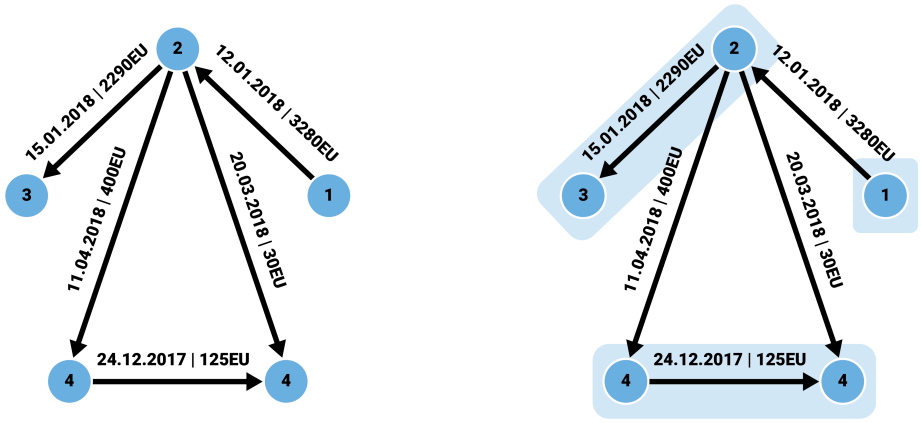


Figure 5.2. An example of the money transfer network. a) The G=(B,E) network so the transfers between the simple bank accounts. b) The hypergraph H=(P,E) where the nodes are the clients and one node can contain multiple bank accounts from the original network. The red rectangles indicate the clients in the H network.

5.3.2 Transaction Cycle Detection

To define transaction cycles, we will apply temporal data networks, where both the edges are attributed with time stamps and transaction data. We also note that the following definition of transaction cycles can be generalized in a natural way both on the time dimension and the related data records: different constraints can be considered for identifying an edge to be "living" from the point of view of a given cycle. For example, we can make restrictions by limiting the (relative) difference of the transaction amounts or the time difference of two consecutive transactions. The developed method can be also easily modified accordingly, and we are using these constraints for demonstration as they were defined by experts of the bank from the use case.

Therefore, let a transaction cycle be a closed walk where:

- The repetition of the nodes along a cycle is allowed, while for the edges it's not allowed
- Let $t_e^0, t_e^1 ... t_e^{n-1}, t_e^n$ be the sequence of the timestamps along a valid cycle where every $t_e^i \leq t_e^{i+1}, i = 0...n-1$ so the timestamps are in ascending order.
- Let a_e^0 be the amount of the first transaction and a_e^i be the amount of the further transactions with $i = 1...n$. Then $a_e^0 \cdot (1-\alpha) \leq a_e^i \leq a_e^0 \cdot (1+\alpha)$, meaning that the difference between the amount of the first transaction and other transactions is bounded by the previously given real value $0 \leq \alpha \leq 1$. The α value is a parameter of the methodology, it is defined by the specific user requirements.

It is easy to see, neither general detection methods nor the elementary cycle detection is able to

consider all of the requirements: in general methods the repetition of the nodes is not allowed, while the elementary cycle detection method is not able to consider the time and the amount constraints. In the general cycle detection the input is a network, in our case it is the hypergraph $H(P, E)$, and the output is the set of the cycles in the network. Table 5.1. shows an example input file with different transaction data.

Table 5.1. Example produced input file where the clients are already connected to their bank accounts.

ClientFromID	ClientToID	IBANFrom	IBANTo	Date	Amount
3	1	IBAN3-1	IBAN1-1	26.09.2018	250 Euro
1	2	IBAN1-1	IBAN2-1	03.09.2018	255 Euro
2	3	IBAN2-1	IBAN3-2	20.09.2018	245 Euro
3	4	IBAN3-1	IBAN4-1	19.09.2018	390 Euro
4	5	IBAN4-2	IBAN5-1	17.09.2018	1000 Euro
5	2	IBAN5-1	IBAN2-2	01.09.2018	768 Euro

The presented input format contains the information about the clients such as the bank accounts, so every client has a ClientID, which encapsulates all of the bank accounts of the client. The data represents 6 different transactions between 5 different clients, where the clients with ClientID 2,3,4 have two different bank accounts. In the defined graph every clientID will be a node, and every line in the input file will be an edge with the actual properties between the clients. The corresponding graph H, where the nodes are the clients and the edges are transactions can be seen in Figure 5.3.

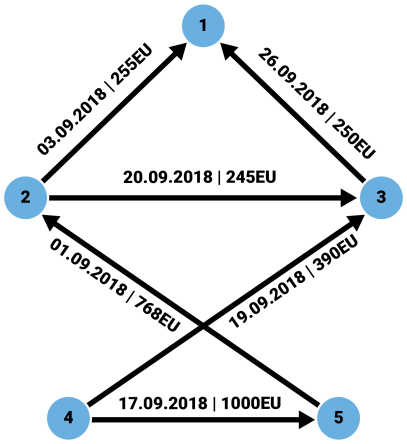


Figure 5.3. An example of a graph which contains transactions cycle. This figure shows the transactions of 5 different clients. Based on the cycle definition, along the 1-2-3-1 closed walk the timestamps are in ascending order and it is a cycle if the $\alpha \geq 0.1$ nevertheless for example the 5-2-3-4-5 is not because of timestamps and amounts.

5.3.3 Special Cases

It is important to detect all cases which fit into the definition because if a cycle remains undetected and the method is not able to recognize a special case, the fraudsters can use this to hide their frauds. The main challenge in our research was to understand the nature of the transaction cycles and define the special and possible cases of the structure. Since the order of the nodes along a cycle is restricted, every cycle can be a valid cycle only from one starting node, if the amounts are within the specified limits. In real life, any type of money movements can be a cycle from our point of view, even multiple back and forth transactions between only two clients. Since our algorithm based on DFS, the special and interesting cases are the ones where two or more cycles are part of each other along transactions or when a client provides a loan for multiple clients. It means that cycles can be part of each other along transactions or nodes, or a node can be repeated through a cycle. Figure 5.4 shows two different examples for the special cases mentioned above.

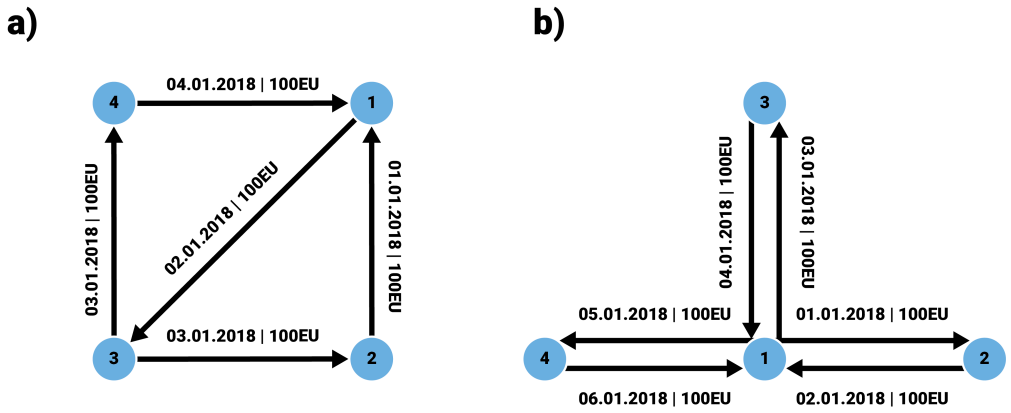


Figure 5.4. Two special cases of the cycle definition. Figure a) shows two cycles (1-3-4-1 and 2-1-3-2) which are part of each other. The starting transaction of the first cycle is 1-3 which realized on 02.01.2018 while of the second cycle is 2-1 on 01.01.2018. This structure can occur with arbitrary number of cycles in any rotations. Figure b) shows the 1-2-1-3-1-4-1 cycle which represents back and forth transactions from client 1 to the others. Naturally any combinations of the special cases can occur in the network.

From theoretical point of view it is important to deal with these cases even if in real life these are not so common: if a possible fraudster knows the detection method, he can take advantage of it, and make a fraud in a way that the method cannot detect. Since it is a structural pattern and our method can detect each case based on the definition, the method is undeceivable.

5.3.4 Method

In most cases bank is partially unstructured (depending on the actual bank); hence as a first step the framework cleans the data and transforms the IBAN numbers into common format in the transaction table and in the client-bank account data. After cleaning the client nodes are produced based on the key value pairs and the hypergraph is built ("hypergraph" in this case refers to the fact that all bank accounts of the same client are combined into one node). As Algorithm 5 shows the input of the method is the hypergraph $H(P, E)$, and the edge list is ordered by the date stamps. The visiting technique sets every edge to "actually" visited in every root call, while the first transaction of the cycle will be "finally" visited: it means that every cycle from the actual ancestor transaction will be identified by the the first edge. After every root call the algorithm sets the actually visited edges to "unvisited", but the finally visited edges will not be visited again by the algorithm. To ensure the correctness of the algorithm, ordering is needed as the DFS will be called in sequence for every edge based on the ordered edge list. The reason for the above solution is justified with the following example: in Figure 5.4. a) case the algorithm finds the 1-3-4-1 cycle first, then the 2-1-3-2 remains undetected since the state of the 1-3 edge will be "finally" visited. Nevertheless, if we call the recursive function based on the ordered edge list, and the algorithm searches the cycles by the date, every cycle can be detected. Algorithm 10 shows the pseudo-code of the detection method.

In the pseudo-code the inputs are the following: $H(P, E)$ is the hypergraph and O denotes the ordered edge list by the date stamps. First, the method initializes set C, which is the set of the detected cycles, v_0 as the ancestor node, e_0 which is the first transaction, and *depth* for the recursion depth. The recursion depth is recorded in every call and it denotes the length of the actual walk in the network. From line 9-22 the recursive function is presented with a node, edge, α, *maxlength* parameters, where n is the actual node wherein the recursion is called, e_0 is the first transaction, α is the bound for the difference of amounts, and the *maxlength* is a limit for the maximal c*maxlength* line 11, a cycle is found if the actual node v is equal to the node v_0 (starting node), the depth is between 1 and the parameter *maxlength*, as well as the length of the cycle is maximal, so it cannot be extended with additional transactions. The $out\_edges(v)$ in line 17 means the out edges of node v:if the recursion depth is greater than one, the algorithm searches the cycles through every unvisited neighbours where the date stamp and the amount of the transaction meets the conditions. In other words, the recursion searches in a direction where the amount and the time parameters are correct. The kernel of the method is the calling order which can be seen in line 28, where DFS is called for every ordered edge in the list O. The output of the algorithm is the set C of detected cycles. The complexity of the algorithm in one iteration is $O(|P| + |E|)$ and in overall case it depends on the number of the cycles in the network. Nevertheless, in the case of real life transaction networks, parameters α and *maxlength* limit the depth of the recursion with providing a reasonable running time. The advantage of the algorithm is that it is able to handle different properties in connection with the transactions, like date stamps, different amounts, or any restriction which can be formalized

Algorithm 10 Cycle detection Method

Input:
$H(P, E)$
O (ordered edge list by datestamps)
Method:
$C \leftarrow \emptyset$
$v_0 \leftarrow \emptyset$
$e_0 \leftarrow \emptyset$
$depth \leftarrow 0$
function CDM(v,e_0, α, $maxlength$)
    $depth++$
    If: c cycle is found so $v == v_0$ where $depth > 1$ and $depth < maxlength$
        $C \leftarrow c$
    If: depth $== 1$
        $e.visited \leftarrow visited_{actual}$
        CDM($e_0.to()$,e_0,α,$maxlength$)
    Else:
        For: $\forall e$ *in* $out\_edges(v)$ where $(abs(e_0.a_e - e.a_e) < \alpha \cdot e_0.a_e$ & $e_0.t_e < e.t_e)$
            If: $(e.visited == unvisited)$
                $e.visited \leftarrow visited_{actual}$
                CDM($e.to()$,e_0, α, $maxlength$)
    $depth--$
End function
For $\forall o$ in O
    $v_0 \leftarrow o.from()$
    $e_0 \leftarrow o$
    CDM(v_0,e_0, α, $maxlength$)
    $e_0.visited \leftarrow visited_{final}$
    For $\forall v$ *in* V
        If $(e.visited == visited_{actual})$
            $e.visited \leftarrow unvisited$
Output: C set of found cycles

as a constraint before the next recursive call.

5.4 Case Study

Our methodology is general with respect to different parameters, however, the demonstrated use as outlined earlier, is defined by a Hungarian bank that introduced a new fraud system.
The fraud detection system monitors the activities of the clients and the administrators. In the system there are more than 50 indicators to indicate the suspected frauds, and one of the key indicators is the transaction cycles using our method. The reason is that the method is potentially integrated with any attribute connected to the edges or the nodes, the methodology can be combined with other filtering procedure defined by the indicators. During the method the constraints for edge

selection can be evaluated dynamically, for example the bound for the amount can depend on the values of other attributes as well.

Transaction cycles (like generally fraudulent actions) are very rare, especially if the length of the actual cycle is longer than 3 and the amounts are closely similar. In this case study real transaction cycles identified by the system presented in a way that the IBAN numbers of the clients are masked, but the dates and the amounts are real. As a case study network, we received transactions of a part of the year of 2016. Te network contains 139242 bank accounts and 1296815 transactions. Table 5.2. shows the results on the real transaction network with different parameters. The bank preferred the $\alpha = 0.1$, $Min\ length = 3$, $Max\ length = 6$ parameters, because in real life the longer cycles could be random transaction cycles also.

We tested our algorithm with 2 different α values and 3 different maximal length values, thus

Table 5.2. Results of the cycle detection algorithm on the real transaction network

Parameters	Number of cycles	Number of bank accounts	Number of clients
$\alpha = 10\%$ $Min\ length = 3$ $Max\ length = 6$	276	440	437
$\alpha = 20\%$ $Min\ length = 3$ $Max\ length = 6$	789	1026	997
$\alpha = 10\%$ $Min\ length = 3$ $Max\ length = 10$	277	449	447
$\alpha = 20\%$ $Min\ length = 3$ $Max\ length = 10$	835	1122	1087
$\alpha = 10\%$ $Min\ length = 3$ $Max\ length = 20$	277	449	447
$\alpha = 20\%$ $Min\ length = 3$ $Max\ length = 20$	839	1148	1111

with 6 different parameter combinations. The minimum length was 3 in each case, because in this case the back and forth transactions were excluded which are not important from the fraud point of view. As Table 5.2 shows, if we allow 10% difference between the amounts along the cycles, the length of the allowed transaction cycles doesn't have too much effect to the number of the founded cycles. The number of the corresponding bank accounts is very low compared to the number of the bank accounts in the whole network. In this case the number of the clients is not higher than the number of the bank accounts, which means that the clients were not using multiple bank accounts

through the cycles. As the table shows in the second, fourth and sixth cases the α has big influence to the number of cycles and corresponding accounts. In these cases some clients were using multiple accounts, which could mean random cycles, or suspected fraud activities also. The running time of the algorithm was within 1.5-2 hours in every case, but in real life applications the running time is not so critical because mostly the bank runs the algorithm only for a shorter period of the year, even though the method is able to handle longer periods (even several years). The complete system has been working at the institute since 2018 and it detects 2-4 suspected fraud activities in every month and it also detected some fraud-related client base.

5.5 Summary

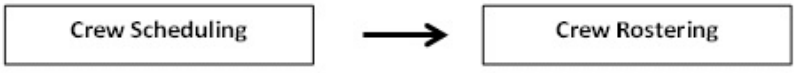
Fraud detection systems play central a role in the life of financial institutions, but it is generally problematic to detect structural patterns in the data. The main challenge is to turn the real processes into the data level, understand the structure and develop a method which is able to detect all of the occurring patterns that are interesting for the financial institute. In this chapter of the dissertation, we introduced a DFS based solution for the cycle detection problem, which is able to detect the real life structures in the network and can handle time, amount and other parameters. The main part of the algorithm is the visiting technique and the ordered edge list based root calling sequence. It is a powerful tool to detect the different types of transaction cycles in the network, because it offers a wide parameterization in which the bank can set their preferences. The research was initiated by a Hungarian bank and the produced method has been used in the real life also since 2018. The methodology also can be used to detect cycles with additional preferences, where e.g. the time between the transactions is limited, or other time or amount based regulations are defined.

Chapter 6

Network Based Crew Rostering

In this chapter of the dissertation, we introduce a use case where a network represents the search space of a real world optimization problem. In certain areas of the industry, the workers' work is performed not in a fixed order. The work activities are organized into shifts, which may vary in duration, time of the day and other properties. In generally, every worker has a contract with a defined expected worktime and a base salary for that, hence the overtime or the undertime is a large-scale extra cost for the company. In the life of the companies, the human cost is a significant part of the complete budget, hence they want to employ the workers in the most efficient way. In most cases, the scheduling of the workers has two different steps (see Figure 6.1.). The first is the crew scheduling which means that the daily tasks are catenated into shifts so that each shift must meet the law constraints. The second step is the crew rostering. In this step the question is how to assign the crew members to shifts for a work period called planning period which is typically being 1-3 months long. This chapter concentrates on the rostering phase. It is important to note, that our solution is a proof of concept, even though it is possible to extend the core of the method with more complicated regulations. The objective of this research is to show the efficiency of the core algorithm with basic regulations which are mostly used internationally as we did not intend to give a country specific solution. In real life applications very specific regulations are applied in several cases, such as the constraint of the travel time from home, or fair distribution of popular and not popular shifts can be also taken into account [10]. In summary, in real life environment the solution for crew rostering must meet some European Union and local regulations as well as the preferences of the workers.

Figure 6.1. Scheduling of workers



In this chapter of the dissertation, we give a new heuristic solution method based on graph coloring which fits to many application area. Our algorithm has two main steps. In the initial rostering, the algorithm estimates the number of workers, builds a conflict graph from the shifts, and generates days-off pattern for every worker. One of the key innovations of the method is to generate days-off patterns which meet the basic regulations and create a frame for the problem, and to get the additional free days indirectly from the tabu search method. Afterwards the graph is colored, so an initial rostering is created by a modified greedy algorithm. When an initial solution is generated the graph is iteratively recolored with a tabu search method to reduce the cost. An important part of the method is to balance the shifts among workers to create a solution where workers are close enough to their expected workload. The results of the algorithm have been compared to the results of the integer programming model with moderate problem size. These results show that our algorithm is efficient and robust. Our solution is a wireframe for the general crew rostering problem, however we tested our method in a real-life application area of the driver rostering case. In the next two sections the crew rostering problem is discussed and defined. In Section 6.3 the mathematical model of the problem is introduced, while in Section 6.4 and 6.5 our heuristic method and the test results will be presented.

6.1 Crew Rostering

There are two different cases concerned in the workers-shifts assignment. In the first case the companies assign an optimal number of crew members to the pre-defined daily shifts minimizing the total cost. In the second case there is a given set of workers and the firm assigns the shifts to them in a most efficient way. In this chapter we deal with the first case i.e. we have shifts and the main goal is to minimize the overall cost.

The crew rostering problem is based on the generalized set covering model. Dantzig was the first who dealt with its mathematical application [42]. The crew rostering problem is NP-hard, therefore it is generally considered that exact solution is not realistic to produce in the case of real life size problems [69, 127, 91, 115, 64]. By the reason above as a consensus of theory and practice, heuristic algorithms are used. The crew rostering solution methods have a quite rich literature, several overviews are available [85, 55, 129, 139, 56, 44]. The literature provides numerous examples of the issue, among which the most significant ones are the airline crew rostering [47], railway crew scheduling [84] and the driver scheduling [9, 94]. In the literature, the studies are generally grouped around the related application areas. These solutions need to correspond to the regulations of the company as well as to the "national norms". In most cases the regulations are different in each area, for instance the qualifications of the workers are handled in different ways in airline crew rostering, while in driver rostering it is usually ignored. These regulations are formalized as constraints which usually have two different types. A hard constraint must not be violated, while in a case of a soft

constraint it is allowed with being penalized by some extra cost. For example single workday in a long days-off period or work in a requested days-off can be handled by soft constraints. We only deal with hard constraints but the model and methodology can be easily extended to soft constraints using appropriate penalties in the objective function.

Taking into consideration the basic regulations of all the significant areas (typically regulated in national laws), we have applied the following constraints:

1. A worker can have maximum one shift in one day.
2. There is a minimum rest time between two shifts.
3. There is a maximum total worktime during one week.
4. There is a minimum number of days-off in one month.
5. There is a maximum number of consecutive workdays.

Every worker has a contract which determines the expected worktime. The difference between this defined worktime and the length of the shifts may produce overtime or undertime. For overtime the worker receives extra payment above the basic salary. However, the basic salary must be paid even in the case of undertime. Therefore the human resource extra cost on the schedules shifts can be optimised through minimizing the overtime.

6.2 Problem Formulation

The crew rostering problem is formally defined as the following. Let C be the set of workers. The set of shifts denoted by S which needs to be carried out. A shift is composed of a series of daily tasks. A shift is defined by its date, starting and ending time in the day, duty time (i.e. the time between the starting and ending time) and the working time. The working time might differ from the duty time since a shift may contain idle periods when the worker does not work. The aim is to assign the crew members to the shifts with minimal cost. Consequently, let $f = S \to C$ be an assignment where the shifts are covered by the workers and exactly one worker is assigned to each shift.

Let $ct(i)$ be the contract type of worker i. The type of the contract defines the prescribed worktime in average for a single workday. Let $aw(ct)$ denote the expected daily working hours by contract type ct. Based on the contract it is possible to calculate the expected worktime in the planning period. For example in our case, every crew member has a contract which defines eight working hours per day. In our model, the type of the contract is a parameter, therefore it is possible

to also deal with different contract types. The expected worktime can be defined in the following way:

$$expected\ worktime(i) = number\ of\ workdays * aw(ct(i)) \quad i \in C \tag{6.1}$$

The basic employment cost is based on workers' expected worktime defined by their contract. So in optimal case every worker will work according to her/his expected worktime. However, in case of working over the expected worktime, employers have to provide extra salary for this overtime. Therefore, the cost is the following:

$$cost = \alpha * overtime + \beta * employment\ cost \tag{6.2}$$

where α and β are pre-defined weights. In a real problem these multipliers can adjust the different costs to the preferences of the company. Following the practice we suppose that the employment cost is proportional to the working hours prescribed by the contract type. Hence, the objective function will consider in the minimization both the overtime and the number of workers. We also assume that the planning period P is fixed (typically 1-3 months) and all the rules having no specified time period (e.g. the average working time) are considered with respect to P; with this approach we follow the practice as well.

6.3 Mathematical Model

Let D be the set of the days, $Week$ the set of the weeks and Mon the set of the months in the given planning period. Consequently D_w^{Week} denotes the days of the week w and D_m^{Mon} is the days of the month m. Meanwhile the length of the planning period (the number of days) is defined by l, and let l_m be the number of days in the month m. The set of the workers is denoted by C, and the set of shifts by S where S_p is the set of shifts on day p. Let SS_{jk} is a compatibility relation between the shifts, where its value is 1 if shift j and shift k can be assigned to the same worker, and 0 otherwise (can be used to define Rule 2). Let WT be the maximum worktime on a week (for Rule 3), WD be the maximum number of consecutive workdays (for Rule 5) and D_p^{WD} be such number of the consecutive days beginning from day p $(WD + 1\ days\ in\ a\ row)$. The minimum number of days-off is denoted by RD (for Rule 4). Let the expected worktime in a month be $aw(ct(i, m))$, where $m \in Mon$ and $i \in C$. In order to minimize the number of workers let $cc(i)$ be the operational cost of worker i, i.e the base salary. Let $wt(j, p)$ be the worktime of shift j on day p. Finally let α be the weight of the overtime and β be the weight of the employment cost. We need the following variables to define the model:

Driver i assigned to shift j:

$$x_{ij} \in \{0,1\} \quad \forall i \in C,\ \forall j \in S \tag{6.3}$$

Driver i works on day p:

$$z_{ip} \in \{0,1\} \quad \forall i \in C,\ \forall p \in D \tag{6.4}$$

Driver i works in the planning period:

$$y_i \in \{0,1\} \quad \forall i \in C \tag{6.5}$$

Overtime of worker i:

$$\pi_i \geq 0 \quad \forall i \in C \tag{6.6}$$

The constraints of the integer programming model are the followings:

Exactly one worker is assigned to each shift.

$$\sum_{i \in C} x_{ij} = 1 \quad \forall j \in S \tag{6.7}$$

There is at most one shift assigned to worker on a given day.

$$\sum_{j \in S_p} x_{ij} = z_{ip} \quad \forall i \in C,\ \forall p \in D \tag{6.8}$$

An employee works in the planning period if he/she has at least one shift.

$$\sum_{p \in D} z_{ip} \leq l y_i \quad \forall i \in C \tag{6.9}$$

The following constraint excludes the possibility of assigning two incompatible shifts to a worker.

$$x_{ij} + x_{ik} \leq SS_{jk} + 1 \quad \forall i \in C,\ \ \forall j,k \in S \tag{6.10}$$

The worktime must not exceed the maximum working time in any week.

$$\sum_{p \in D_w^{Week}} \sum_{j \in S_p} x_{ij} wt(j,p) \leq WT \quad \forall i \in C,\ \forall w \in Week \tag{6.11}$$

A worker can have at most the maximum number of consecutive shifts.

$$\sum_{p \in D_q^{WD}} z_{ip} \leq WD \quad \forall i \in C,\ \forall q \in D \tag{6.12}$$

At least the required number of days-off have to be given for each worker in a month.

$$\sum_{p \in D_m^{Mon}} z_{ip} \leq l_m - RD \quad \forall i \in C, \ \forall m \in Mon \tag{6.13}$$

Let us define the overtime.

$$\sum_{m \in Mon} (\sum_{p \in D_m^{Mon}} \sum_{j \in S_p} x_{ij} wt(j,p) - aw(ct(i,m))) \leq \pi_i \quad \forall i \in C \tag{6.14}$$

The objective function minimizes the sum of the weighted overtime and the operational cost.

$$\min \sum_{i \in C} (\alpha \pi_i + \beta y_i cc(i)) \tag{6.15}$$

The mathematical model is a good approach if the problem is relatively small because it gives an exact solution for the problem. Unfortunately, in real life there are often too big problems for these solution methods, therefore in most cases the companies use heuristics.

6.4 Heuristic Method

Our algorithm is a two-phase graph coloring method (TPC), where in the first step an initial rostering is produced with a graph coloring procedure and in the second step this rostering is improved with the over- and undertime being minimized by tabu search. M. Gamache et. al [67] developed a method, where graph coloring algorithm based tabu search was used for scheduling pilots. These pilots have qualification and the objective is to find a feasible solution in a pre-specified workload interval. Nevertheless, in the literature tabu search method is a widely used technique especially in bus driver scheduling or rostering. Some example papers using tabu search to solve the scheduling problem (defining the shifts) are [34, 160], but the method was also successfully applied for laboratory personnel scheduling problems [11]. However, in our case both the problem and the solution method are different by two reasons: the regulations are different and here days-off patterns are used. Furthermore, we will give a general solution method for the crew rostering, serving as a framework to be specialized to several fields. In our approach by its universality and flexibility, days-off patterns will be applied with a k-coloring algorithm.

The TPC has the following steps:

1. Initial rostering.

a. Estimate the number of workers.

b. Generate days-off patterns.

c. Build a conflict graph.

d. Color the graph.

2. Tabu Search to improve the solution.

The initial rostering determines the number of workers, the days-off patterns and the initially colored graph. First, the algorithm estimates the optimal number of workers and for each worker generates a days-off pattern. These patterns will define whether a worker can work on a day or not. In the last two steps of the initial phase the conflict graph is built and colored. In most cases the cost of the initial rostering can be improved, therefore in the second phase the method switches the shifts between the workers with a local search method in order to minimize the cost of the rostering.

It is important to note that in our heuristic the objective function is divided into two parts (see Formula 6.2). The algorithm tries to minimize the number of workers in the initial phase, while the recoloring step is to minimize the overtime. A feature of the heuristic that we get better solution if we also minimize the undertime: we will see below that it is an equivalent approach with respect to Formula 6.2 and it will provide a solution in which the workload will be balanced among the employees.

Nevertheless, the general cost function of the method is based on the overtime and undertime only. Since the employment cost is proportional to the working hours by contract (expected worktime), we will see that Formula (6.2) is equivalent to the weighted linear combination of overtime and undertime. Formally, the overtime and the undertime definition and the cost of the heuristic are the following.

$$overtime = \sum_{i \in C} \max\left(0, worktime(i) - expected\ worktime(i)\right) \tag{6.16}$$

$$undertime = \sum_{i \in C} abs(\min\left(0, worktime(i) - expected\ worktime(i)\right)) \tag{6.17}$$

$$cost = \alpha' * overtime + \beta' * undertime \tag{6.18}$$

In order to clarify the equivalence of Formulas (6.2) and (6.18), notice that the total worktime required by the shifts (the sum of the worktime of all the shifts) in the whole planning period is a constant. Therefore it is easy to see that in any solution the expected worktime is equal to $(total\ worktime - overtime + undertime)$. By this we obtain for Formula (6.2):

$$cost = \alpha * overtime + \beta * (total\ worktime - overtime + undertime) \tag{6.19}$$

Then $(\beta * total\ worktime)$ is a constant and provides the basic theoretical lower bound in the cost. Therefore $\alpha' = \alpha - \beta$ and $\beta' = \beta$ will make Formulas (6.2) and (6.18) equivalent from optimization

point of view with (6.18) expressing the extra cost above the basic theoretical lower bound.

6.4.1 Initial Rostering

Estimate the Number of Workers

At first, the initial set of workers (C^*) needs to be defined wherewith the shifts can be effectively covered. The number of these workers can be estimated for the planning period using the total worktime of the shifts and the contracts of the workers. In our case we concern "uniform workers" meaning that they have the same type of contracts, i.e. their expected worktime is the same. Basically, we suppose that the planning period is one or a few months (typically 1-3 months). This assumption is general in real life situations, but it can be easily relaxed to any length of the planning period. Therefore, the number of the crew members for a given contract type ct is given by the following way:

$$|C^*| = round\left(\frac{total\ worktime}{aw(ct) * number\ of\ workdays}\right)$$

Practically, an initial set of workers is given in such a way that the expected worktime is assigned to all of the workers determined by the contract type in every day of working. In this way, the estimated number will be the cardinality where the difference of the total worktime and the expected worktime for this set is minimal.

A trivial lower bound for the number of workers can also be given by the number of the shifts on the busiest days, since one crew member can have at most one shift a day. Furthermore, if the estimated number of workers is known, a theoretical lower bound (LB^{ot}) for the overtime can be given by the following way:

$$LB^{ot} = \max\left(0, (total\ worktime - |C^*| * aw(ct) * number\ of\ workdays)\right)$$

It is clear that with a given set of workers LB^{ot} is correct, since the overtime is minimal in this case if every worker has workload at least according to his/her contract.

Generate Days-off Patterns

In some areas, it may be necessary to pre-specify days for a worker in advance when he or she does not work (called days-off) in the planning period (called days-off pattern). Some example for the usage of days-off patterns can be found in the literature [130, 52]. It is a widely accepted and used methodology to generate a days-off for the workers in the crew rostering. The advantage of the days-off patterns is that the days-off requested previously by the workers can be taken into account, as well as the 4th (minimum number of days-off in one month) and 5th (maximum number of consecutive workdays) rules are fulfilled here, so we do not have to concern these in the latter phases.

Days-off patterns define different fixed free days for each worker. We tried 3 different methods to generate the days-off patterns. At first we generated random patterns as presented in Table 6.1. The days-off are generated by weighted random generator which consider the number of the shifts on the days and Rule 4 and Rule 5. On days with higher shift load the workers received a day off with lower probabilities than those days where lower number of shifts are. To formalize the probability let $|C*|$ be the estimated number of workers and S_j denote the set of the shifts on the day j, the probability of the days-off on the day j is given by $1-(|S_j|/|C*|)$; if the generated days-off pattern doesn't meet with the rules, we insert additional free days into the pattern randomly. The pattern vector is generated for every worker one by one.

Table 6.1. Random days-off patterns

Worker	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1.	0	1	0	0	1	0	1
2.	1	0	1	1	0	0	0
3.	1	0	0	0	0	1	0
4.	0	1	0	0	1	0	1
5.	0	0	1	0	0	1	0
6.	0	0	1	0	1	0	1
7.	0	0	1	0	0	1	0

The second tested pattern type was the so called 5-2 pattern (see Table 6.2). Here, the workers get two days-off after each consecutive five workdays. These days-off patterns repeat a seven days long sub-pattern along the planning period such that if the sub-pattern starts at day j^* then from day j^* to day j^*+4 the days are workdays and days j^*+5 and j^*+6 are days-off. The next pattern shifts the sub-pattern forward with one day. Therefore, seven different patterns are generated from this days-off pattern type.The 5-2 pattern seemed to be appropriate, since a week usually contains 5 workdays and 2 days off. Hence, if everyone will work by this 5-2 pattern, then their working hours are likely close to their expected worktime. It is clear that the 5-2 days-off patterns also meet the defined hard constraints.

We found that the random and the 5-2 patterns are too rigid and greedy with producing too much exclusion in the first phase. This means that during the graph coloring the days-off patterns exclude too many workers on days where they could possibly work.

To overcome this problem the third pattern type defines minimal fixed days-off considering the rules. Thus, we propose a 6-1 days-off pattern which means that the workers get one fixed days-off after every six consecutive workdays. These patterns are generated with the same method as 5-2 patterns, but in the sub-pattern from days j^* to j^*+5 are workdays and day j^*+6 is days-off (see in Table 6.3). This pattern meets both the minimal free days rule (Rule 4) and the maximal consecutive workdays (Rule 5) based on the European regulations (see in Section 5). It causes only a few exclusion during the graph coloring. Also, the tabu search will have a relatively big state space.

Table 6.2. 5-2 days-off patterns

Worker	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1.	0	0	0	0	0	1	1
2.	0	0	0	0	1	1	0
3.	0	0	0	1	1	0	0
4.	0	0	1	1	0	0	0
5.	0	1	1	0	0	0	0
6.	1	1	0	0	0	0	0
7.	1	0	0	0	0	0	1

Naturally, the pattern defines only fixed days-off and in the last phase we may get automatically some additional days-off in the final solution: when the workers are not assigned to a shift they automatically have a day off. Because of the days-off patterns, the 4th and 5th rules does not need to be taken into account later in step 2. In the Table 6.2 and Table 6.3 it is enough to give the day-off patterns for the first seven workers, since if i and j are the indexes of the workers, and positive integers, furthermore $i \equiv j \ mod(7)$ then the $i-th$ and $j-th$ workers have the same days-off pattern.

Table 6.3. 6-1 days-off patterns

Worker	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1.	0	0	0	0	0	0	1
2.	0	0	0	0	0	1	0
3.	0	0	0	0	1	0	0
4.	0	0	0	1	0	0	0
5.	0	0	1	0	0	0	0
6.	0	1	0	0	0	0	0
7.	0	0	0	0	0	0	1

Build a Conflict Graph

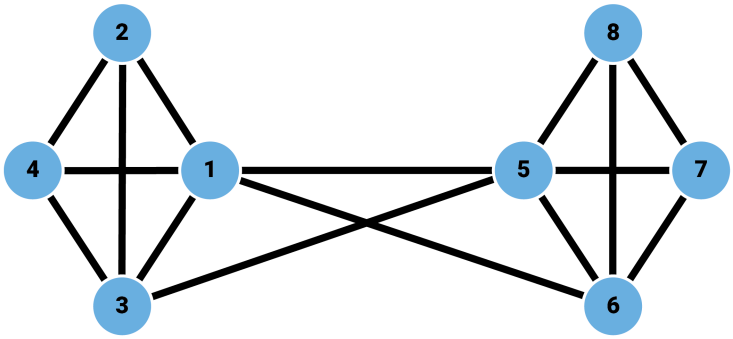
Let $G = (V, E)$ be a graph so called conflict graph, where every shift in the planning period is a vertex of the graph. There is an edge between two vertices if the corresponding shifts must not be performed by the same worker. For example if the time between two shifts is less than defined by the 2nd rule or they are on the same day.

To understand the graph building let us see an example. Let the initial input of two days with 8 shifts is prescribed in Table 6.4. The shifts of each day compose cliques. There are 3 additional edges between them (1-5,1-6,3-5) because the time period between these shifts is less than the required one by Rule 2 (general value is 12 hours what we use here too). The generated graph is presented by Figure 6.2.

Table 6.4. Example shifts with their date,startin and ending time in minutes (0-1440 a day), and the contained working time in minutes.

Id	Date	Begin	End	Worktime
1	2016.01.01	889	1411	522
2	2016.01.01	540	984	444
3	2016.01.01	714	1135	421
4	2016.01.01	237	713	476
5	2016.01.02	396	850	454
6	2016.01.02	454	992	538
7	2016.01.02	702	1138	436
8	2016.01.02	814	1327	394

Figure 6.2. Example graph corresponding to two consecutive days.



Color the Graph

Node coloring of a graph means assigning a color to each node in such a way that every neighboured node has different color. A coloring of the conflict graph gives a rostering for the problem if one color corresponds one worker. This rostering is correct since it fulfills the defined rules: Rule 1 and Rule 2 are guaranteed by the structure of the conflict graph, Rule 4 and 5 are fulfilled by the days-off pattern and Rule 3 is handled in the coloring. If we set the ending time of the vertices to the maximum of the increased value by the time of the 2nd rule and that of the end of the day, then we will obtain an interval graph [82]. There is an efficient algorithm to color interval graph [72], but due to the extra restrictions of the crew rostering problem it becomes NP-hard. A good overview of the problem written by Kolen et. al [13]. Such a k-coloring algorithm is needed while adding a new color is possible. The initial estimation of the number of workers is usually correct, but some extraordinary inputs, e.g shifts with too short worktime, result such a graph that could

not be colored with the estimated number of colors. Therefore we used the DSATUR algorithm which is effective and can be easily adopted to our needs (see Brelaz [21]) with coloring the nodes by their saturation values (saturation of a vertex represents the number of different color classes in its neighbourhood). Each coloring iteration chooses the node with the highest saturation value and chooses the color where the corresponding worker is in maximum undertime. It starts with the estimated number of colors (i.e worker) and adds new color if it is necessary (i.e no available worker for a shift).

Algorithm 11 Graph coloring.

1: **While there is an uncolored vertex**
2: Let v be the next uncolored vertex with the maximal saturation value
3: If there is an equal saturation we choose the vertex with maximal degree
4: $C \leftarrow$ available colors where the regulations are not violated
5: Choose available color $c \in C$ (i.e worker) with the lowest worktime.
6: If c does not exist let c be a new color.
7: Assign c to v.
8: **for** each $v_n \in neighbours\ of\ v$ **do**
9: $v_n \leftarrow update\ saturation\ value$
10: **end for**
11: **end while**

The graph coloring gives an initial solution which meets the given constraints and assigns a worker to every shift. Since the real working time of each worker and their expected worktime is known, the initial cost can be calculated.

6.4.2 Tabu Search

The tabu search was introduced by Glover in 1986 and it is one of the most famous local search techniques [71]. The state space of the tabu search denoted by SL consists of all the feasible coloring patterns in the conflict graph. Each state is a coloring and the objective function has to be minimized on SL. The algorithm visits solutions $sl_0, sl_1, ...sl_n \in L$, where sl_0 is the initial solution produced by the graph coloring and $sl_{i+1} \in N(sl_i)$ with $N(sl)$ denoting the set of neighbours of sl. The next neighbour is chosen by a first fit method meaning that the first neighbouring solution is chosen being better than the actual solution and the neighbours are in a random order. Steps chosen in one step are stored in the tabu list denoted by TL. The steps in the list are forbidden to repeat. The neighbours of a solution sl are defined by using the following operations:

1. *Recoloring of a vertex:* Another color is given to a vertex of the graph. It means that a shift is taken away from a worker and given to another one permitted by the constraints.

2. *Swapping the colors of two vertices:* The colors of two vertices are switched, therefore shifts are swapped between two workers in a way that both of them receive a shift which they can

carry out with keeping the constraints.

The algorithm tries to change or swap a shift between the colors of the most undertimed worker and the most overtimed worker. After carrying out the vertex recoloring or the vertex swapping, the step is added to the tabu list. So the moves are stored in the tabu list instead of the states. The elements spend a specified number of iterations in the tabu list, and if the length of the tabu list is greater than the max allowed iteration for an element on the list, then the oldest element which took the longest time on the TL, will be deleted from it. Pseudocode is given in Algorithm 12.

Algorithm 12 Tabu search.

1: $s_0 \leftarrow$ *initial solution*
2: $TL \leftarrow \emptyset,\ s \leftarrow s_0,\ best \leftarrow s_0$
3: **while** *stopping criteria* **do**
4: $E \leftarrow$ *workers ordered by worktime*
5: **for** each $e_1 \in E$ *from undertimed to overtimed order* **do**
6: **for** each $e_2 \in E$ *from overtimed to undertimed order* **do**
7: $s* \leftarrow$ *the first recoloring or swap between* e_1 *and* e_2 *where* $cost(s*) < cost(s)$ *and* $move(s, s*) \notin TL$ and goto line 10
8: **end for**
9: **end for**
10: **if** $s*$ *not exist* **then**
11: $s* \in N(s)$ *the first solution where* $move(s, s*) \notin TL$
12: **end if**
13: **if** $cost(best) > cost(s*)$ **then**
14: $best \leftarrow s*$
15: **end if**
16: $TL \leftarrow TL \cup move(s, s*)$
17: **if** $TL.size >$ *max size of tabu list* **then**
18: Remove the oldest element from TL which took the longest time on the list
19: **end if**
20: $s \leftarrow s*$
21: **end while**

Multiple methods can be applied as stopping criteria. In most cases, the criteria is based on the iteration number or on the time bound. In our case, the algorithm stops if it cannot find a better solution within a certain number of iteration.

6.5 Test Results

In this section we will introduce the specific regulations for our case study, the test cases and the results of our algorithm. The algorithm was implemented in Java language, and IBM Cplex 12.4 software was used to solve the integer programming part (see Section 6.3). The heuristic was run on the following computer:

- Processor: Intel Core I7 970 3.2Ghz
- Memory: 14Gb.
- Operation System: Microsoft Windows 7 Enterprise 64bit

Since the problem definition ignores the personal conditions i.e. illness or holiday, usually the rostering method is a part of a decision support system and the members of the crew are fictive workers. One roster is just the duty which can be performed by one worker following the rules. Therefore, the rostering problem deals with driver types. In this research, one driver type is used but it can be extended several different worker types. Although the presented method is general for any crew rostering problem, we tested it with rostering of bus drivers of a city public transport company. In our case every driver had a contract of 8 hours of worktime with respect to an average workday. The planning period (the timeframe with respect to which the average daily working time should be 8 hours) was 1 month. The following regulation constraints were used for our test cases, as being the general rules in the driver rostering area:

- The minimum rest time between two shifts lasts for 12 hours (Reg. 2.)
- The worktime must be less than 48 hours in a week (Reg. 3.)
- Drivers must have at least 4 free days in a month (Reg. 4.)
- The number of the maximum consecutive workdays is 6 (Reg. 5.)

The stopping criteria of the algorithm is given in a way that it ends when no improvement in solutions is found after 200 iterations, or when drivers with under- or overtime left only. In the objective function of the mathematical model $cc(i)$ was defined by the expected worktime, and we set $\alpha = 2$ and $\beta = 1$. We evaluated the methods with Formula (6.18) during the testing, where $\alpha' = 1$ and $\beta' = 1$ by the notes following the definition of the formula. We have chosen this case as the typical example when overtime dominates the cost function. The use of parameters α and β makes our approach flexible: if we consider the above natural case ($cc(i)$ being the expected worktime), then practically $\beta = 1$ and $1 \leq \alpha \leq 2$. Since our goal is to show the applicability of the method as a proof of concept, we made our testing on the above basic case.

The algorithm was tested on real-life problems and on randomly generated data. The real-life instances came from the bus driver scheduling department of the local city bus company Szeged, a mid-size city of Hungary. The artificial problem instances are generated in such a way that the properties of the shifts reflect the characteristics of real life data. The shift distributions of the generated samples are the same with the following properties of the generated shifts:

- Worktime is between 7 and 9 hours.
- Duty time is between 6 and 10 hours.

- The division of the shifts on a day follows the normal distribution.

The tests on the generated inputs were running in five problem groups of different size and we also tested our algorithm on a real world input (25,50,75,100,200,real). In this context by the size we mean the number of shifts on the most loaded days of the planning period. The workdays are considered with the maximal number of shifts, while weekends are generated with approximately 80-90% of this maximum. The time limit of the CPLEX was 8 hours in each case.

Table 6.5. Test results on size (5, 50, 75, 100, 200, real)

Input	Lower bound	Cost heur	Time heur	Cost IP	Time IP
gen25_1	4807	4807	3.2	4900	18.77
gen25_2	4658	4658	3.6	4691	495.2
gen25_3	1825	1825	4.1	1855	154.8
gen25_4	4371	4371	3.1	4457	22.96
gen25_5	4815	4815	5.4	4911	18.11
gen50_1	3359	3359	5.4	3415	875.1
gen50_2	2581	2581	15.2	2581	844.8
gen50_3	3000	3000	10.4	3047	1102.34
gen50_4	1671	1671	4.8	1696	1252.78
gen50_5	1925	1925	10.3	1960	1529.56
gen75_1	2861	2861	50.9	-	-
gen75_2	4051	4051	65.1	-	-
gen75_3	3809	3809	88.9	-	-
gen75_4	90	100	79.4	-	-
gen75_5	4682	4682	65.3	-	-
gen100_1	3261	3361	124.4	-	-
gen100_2	987	2581	145.2	-	-
gen100_3	1663	3000	160.4	-	-
gen100_4	1671	1671	121.8	-	-
gen100_5	1925	1925	130.3	-	-
gen200_1	3564	3564	420.4	-	-
gen200_2	4895	4895	321.2	-	-
gen200_3	3771	3771	345.4	-	-
gen200_4	2930	3930	329.8	-	-
gen200_5	2029	2029	298.3	-	-
volan_real	4897	4897	9.87	5012.87	21418.56

The results of TPC are compared to the previously defined lower bound and the solution of the IP, see Table 6.5. It can be stated that in most cases our algorithm produced low running time and good solutions for each test case. The time complexity is found much higher for the IP than the heuristic in all cases. Though heuristic can not guarantee optimal solution, but in practical situations the running time is also an important aspect. However, as the problem size increased the running time of IP became unacceptable.

As the test cases have shown, the iteration number was high enough to reach the lower bound in most cases. The IP was running with a relative gap of 2% in a case of the generated input and 4% in a case of the real input which means the IP stopped when the actual solution was maximum of 2% and 4% distance from the optimal solution. The above fact explains why the costs of the IP are slightly higher than that of the TGPM.

The real input consisted of approximately 4000 shifts in a one month planning period, i.e. approximately 120-140 shift per each workday. The number of the working drivers in the final solution was 150. The results of the real input were tested with the integer programming model, the results of the IP can be found in the last row of the table. With the TPC the problem could be solved with the estimated number of drivers in every case. We obtained that TPC is able to handle relatively large inputs producing good quality feasible solutions in reasonable running time. Furthermore, in the majority of the test cases, our method has reached the theoretical lower bound.

6.6 Summary

In this chapter of the dissertation, important aspects of the crew rostering in the scheduling of bus drivers were introduced. First a mathematical model of the problem was defined. Then we proposed the TPC method for the crew rostering problem, which is a proof of concept dealing with some general international regulations. In the initial phase a preliminary rostering is constructed. First the number of workers is estimated, then days-off patterns are generated, and a conflict graph is built. Lastly, to produce an initial rostering, the graph is colored with the estimated number of colors, where each color refers to an worker. In the second phase, a tabu search method recolors the graph to reduce the cost by minimizing the total undertime and overtime. Our method has been tested with artificially generated and real life instances. The real instances belongs to the local public transport company of Szeged. For moderate sized problems, the results of the presented algorithm have been compared to the solutions of the appropriate integer programming model and to a lower bound. TPC produced a satisfactory running time, reached the lower bound in most cases and returned a feasible solution in all cases. In the future the running time could be improved with other methods such as paralell programming and additional regulations could be taken into account in the conflict graph with additional edges, or with penalties in the objective function.

Chapter 7

Summary

7.1 Summary in English

The objective of this dissertation is to introduce novel network based data analytical and optimization methods as well as the work of the author in this area. The work discusses the topics in an application-oriented way, most of which are strongly related to real life problems. The dissertation can be divided into five major topics which are the following:

1. Development of community detection and infection modelling methodology on public transportation networks.
2. Analysis of the connection between community detection and infection maximization as well as the development of new greedy based infection maximization algorithms.
3. Development of an infection minimization method for the optimization of targeted interventions.
4. Development of a financial fraud detection algorithm on temporal networks.
5. Development of a network based optimization method to solve the crew rostering problem.

Thus, the dissertation can be divided into six chapters, where the first is an introductory part of our work which defines the basic definitions that are needed to understand the thesis points and gives a short historical overview connected to the topic of the dissertation.

After the introduction in the second chapter, a methodology was introduced which can be used in public transportation networks [78]. The method is able to reveal the travel patterns and the fine hidden connections between the passengers, as well as to identify the vulnerable parts of the public transportation system in a case of an epidemic. We tested the method on the public transportation system of Minneapolis using the co-travelling network from the FAST-TrIPs travel demand model

[100]. We introduce the obtained results taking into account the geographical conditions of the city and the habits of the passengers.

In the third chapter, we further examined the infection processes and presented a methodology related to the topic of community detection and infection maximization [76, 77]. In one hand, the method is intended to increase the efficiency of the original greedy algorithm by reducing the search space, in the other hand the obtained results provide a methodology to compare and benchmark overlapping community detection algorithms from infection point of view. In the chapter, we tested our method using eight undirected community detection algorithm from the literature, as well as examined the base concept on directed networks.

In the next fourth chapter, a method was presented which is able to minimize the infection based processes in networks using targeted interventions [120]. The optimization model is an extension of the Generalized Independent Cascade where the objective is to find a set of elements with a given size where the reduction of the individual infection levels maximizes the reduction of the global infection value. We introduced the method with a psychological use case where the objective was to reduce the negative well-being in institutions and companies using targeted psychological interventions. We used real data for the testing of the model, as well as psychological aspects were taken into account in our model.

The fifth chapter aimed to present the application of networks in corporate and banking area using a new fraud detection algorithm in temporal bank transfer networks [80]. The method is able to extract an important pattern from a network where the actors may be closely related to various frauds. We tested our method on real anonymized bank data from 2016 including the discussion of the special cases of the pattern. The method is the part of a real fraud detection system of a Hungarian bank since 2018.

In the sixth chapter, we introduced a network based method where the search space of the problem is represented as a graph. The objective of the problem was to solve the optimal employment of the workers at a company so that the solution meets with the different regulations and it is optimal in the term of costs. The method was tested on real and artificially generated inputs and the obtained results were compared to the results given by the Integer Programming model in the case of small inputs.

7.1.1 Community Detection and Infection Modelling on Public Transportation Networks

After the introduction, we started the second chapter of the dissertation with our first methodology which is able to detect travel patterns in public transportation networks as well as to stress test the whole system from epidemiological point of view. [78]. In the first part, in order to make the method easier to understand, we presented the FAST-TrIPs travel demand model [100] and structure of our input network. Afterwards, we defined our method, which can be divided into four essential steps

that are the following:

- The public transportation network is divided along the vehicle trips by the method.
- Maximal cliques are detected using Bron-Kerbosch algorithm [24].
- The transfer network F is created by the method.
- The community network H is created by the method.

As it can be seen, the method creates two additional graph from the initial contact network which contains information about the co-travelings of our passengers. The first network (transfer network) is intended to express the movement between different passenger groups, providing a good base to detect frequent vehicle trip changes and track the movement of these groups. The second (community network) represents the hidden connections between the passengers. Therefore, this representation is suitable for detecting communities of closely related passenger groups.

Epidemic Spreading Risk Application

The hidden connection structure that is given by the method provides an environment where we can examine the possible virus spreading scenarios on the vehicle trip level. In the last part of the chapter, we examined the different infection simulation scenarios using the SI compartmental model. The output of our method is the ordering and evaluation of vehicle trips from epidemiological point of view. In the end of the chapter, the method was evaluated on the public transportation network of Twin Cities Minneapolis.

7.1.2 Infection and Communities

In the third chapter, we further examined the connection between infection maximization and overlapping community structure in general networks. After the introduction of the basic infection maximization models and the Independent Cascade, we introduced different reduction methods that are able to reduce the search space of infection maximization model using the results from community detection methods. The nodes of the network are ordered based on their infectivity score using the so called community value, and the infection maximization method used the top X percent of the $G(V*)$. The basic idea of the community value is to find nodes that are represented in several communities, ensuring the connection with as many dense subgraphs as possible to spread the infection. The steps of the method are the following:

- Detection of overlapping communities in the network.
- The method computes the community values.
- Reduction of the search space based on the community values.

- Infection maximization with the greedy method using the previously defined reduced search space.

The method we proposed, is also suitable to compare and benchmark overlapping community detection methods.

Infection maximization and benchmarking of undirected overlapping community detection methods

To test our methodology, we compared the results of eight different undirected overlapping community detection methods with the results of the original greedy method [95] on randomly generated networks that were generated using the Lancichinetti and Fortunato algorithm [111]. Based on our results, the Infomap [155] community detection method gave the best results from infection maximization point of view. In general, it can be stated that our community value based methods are able to solve the infection maximization problem even on larger networks.

Directed Hub Percolation

After testing the undirected methods, we introduced the directed version of an existing undirected community detection algorithm [28]. Our objective was to further examine the role of different community related structures in infection maximization problem. In this subchapter, we extended the basic idea by presenting the hub value, which tries to determine the value of the node based on the number of the connected directed cliques. In the directed case, we tested our method on random and real world networks. Regarding to the testing, it can be said that the methodology proved to be efficient on both cases.

7.1.3 Uplift Network Model for Targeted Interventions

In the fourth chapter, we presented an Uplift Network Model that can be used to reduce negative processes on networks using optimized targeted interventions. The method extends the concept of infection maximization on the Generalized Independent Cascade model[29] in a way that we maximize the gap from a reference simulation to decrease the global infection of the network. The entities have two initial infection value depending on whether they get an intervention or not. Each node of the $G(V, E)$ network therefore has a $v^{apriori}$ and a v^{uplift} value. The greedy algorithm looks for a set where the intervention on the nodes so the change from $v^{apriori}$ to v^{uplift} probability minimizes the global infection of the network.

Psychological Use Case

In the second part of the chapter, we presented a psychological application of our method where the goal was to reduce negative psychological effects on a social network. During the testing, we

examined a social network that was based on collected data from 14 Norwegian nursing homes and used the WHO-5 questionnaire [172] to measure the psychological level of the employees. Results show that targeted psychological interventions may increase the global mental well-being compared to the random intervention strategy.

7.1.4 Temporal Network Analytics for Fraud Detection in the Banking Sector

In the fifth chapter, first, we introduced the basic pillars of the fraud detection and defined the base model that is needed in our method, as well as introduced our algorithm which is able to detect transfer cycles in temporal networks. Regarding to temporal transfer cycles it is expected, that the timestamps have to be in ascending order and the difference between the amount of the first transaction and other transactions is bounded by the previously given real *alpha* value. The steps of the algorithm are the following:

- The method creates the $H(P, E)$ network where the accounts are grouped along the clients.
- Creating a time-ordered edgelist.
- The method runs a Modified Depth First Search based on the time-ordered edgelist.
- Collecting the transfer cycles that meet with the parameters.

During the chapter, we examined the special cases of the structure to show that the main features of our pattern are significantly different than the features of general cycles.

Use Case on a Bank Transfer Network

We examined the method on the transfer network of a Hungarian bank, and the algorithm was efficient both in terms of running time and detecting the defined patterns. The presented algorithm has also been part of the fraud detection system of the Hungarian bank since 2018 and has successfully detected several fraud attempts during its operation.

7.1.5 Network Based Crew Rostering

In the last chapter, after a brief overview of the topic of crew scheduling and rostering, we introduced a two-step graph coloring algorithm to solve the general crew rostering. First, we described the general problem definition and the mathematical model of the problem then introduced our heuristic method. The steps of our algorithm are the following:

1. Initial rostering.

a. Estimate the number of workers.

b. Generate days-off patterns.

c. Build a conflict graph.

d. Color the graph.

2. Tabu Search to improve the solution.

The method was tested on artificially generated and real-world inputs, and the results of our method were compared to the results of the Integer Programming Model in a case of small networks. Generally, it can be said, that our heuristic solution reached the theoretical lower bound in most of the cases and produced satisfactory results in terms of running time.

7.1.6 Future Work

There are many possibilities to extend the discussed topics of this dissertation. The most obvious approach is to adapt the methods in many different application areas, however, the further development of the discussed methodologies also has a great potential.

Connected to the model that was presented in the first chapter, it may be worthwhile to develop a combined optimization method in public transportation networks that is taking into account the epidemiological aspects regarding to the particular vehicle schedules. An another possible future work is the rescheduling of existing public transportation systems from epidemiological and passenger comfort point of view.

Regarding to the community structure and infection models, we think that the connection of these two areas may have additional benefits in the future. Using the natural connection between dense subnetworks and the spread of different infections, the idea can be exploited in many areas in order to develop new methods and methodologies that can have global effect on a network using the community related local patterns.

The method presented in the third chapter can be used to reduce the effect of negative processes occurring in real life. We plan to extend and apply the model to epidemiological and financial areas in the future. We think that if we use the existing knowledge from the area of infection maximization to reduce global effects, the approach can have many benefits and potential for the future.

In connection with the fourth chapter, our future plans include the introduction of a model where different types of graph motifs and patterns can be can be arbitrarily defined as a parameter of the method. The resulting framework can provide a general method that is able to detect special local patterns and structures defined by the user.

The graph based optimization method that we defined in the last chapter, can become a flexible general base model of the crew rostering problem. Our model uses a problem representation that may provide a good environment where it is possible to extend the method to other areas as well as to handle additional constraints in the future.

7.2 Summary in Hungarian

A disszertáció célja új, hatékony hálózatalapú adatelemző és optimalizáló módszerek bemutatása és a szerző ezen területen végzett munkásságának összefoglalása. A dolgozat szemléletét tekintve alkalmazásorientált módon mutatja be a tárgyalt témaköröket melyek nagy része szorosan kapcsolódik valóéletbeli problémákhoz. A disszertációt öt nagyobb témakörre lehet bontani melyek a következők:

1. Közösségkereső és vírusmodellező módszertan kifejlesztése tömegközlekedési hálózatokon.
2. A közösségkeresés és fertőzésmaximalizálás kapcsolatának vizsgálata valamint új, mohó alapú fertőzésmaximalizáló algoritmusok kifejlesztése.
3. Fertőzésminimalizáló módszer kifejlesztése célzott beavatkozások optimalizálására.
4. Pénzügyi csalásdetektáló algoritmus kifejlesztése temporális utalási hálózatokon.
5. Hálózatalapú optimalizáló módszer kifejlesztése munkaerőkiosztás feladat megoldására.

A disszertáció így hat fejezetre osztható, melyek közül az első egy bevezető rész, mely definiálta a tézispontok megértéséhez szükséges alapdefiníciókat, valamint rövid történelmi áttekintést adott a dolgozat témájával kapcsolatban.

A bevezető után a második fejezetben egy, a tömegközlekedési hálózatokon használható módszertan került bemutatásra [78]. Az eljárás képes az utazási minták és utasok közötti finomabb kapcsolatok felderítésére, valamint a tömegközlekedési hálózat vírusterjedési szempontból sebezhető részeinek detektálására. A módszer tesztelését Minneapolis városának tömegközlekedési rendszerén a FAST-TrIPs travel demand modell [100] által adott együttutazási hálózaton végeztük. A kapott eredmények a város földrajzi viszonyait és az utasok szokásait figyelembe véve kerültek bemutatásra a disszertációban.

A harmadik fejezetben a fertőzési folyamatokat vizsgáltuk tovább, valamint egy a közösségkereső és fertőzésmaximalizáló algoritmusok témaköréhez kapcsolódó módszertan került bemutatásra [76, 77]. Egyrészről a módszer a mohó algoritmus hatékonyságát hivatott növelni a keresési terének csökkentésével, másrészről az így kapott eredmények lehetőséget adnak átfedő közösségkereső algoritmusok fertőzésterjedési szempontból történő összehasonlítására illetve benchmarking módszertan felállítására. A fejezet során a módszerünket nyolc, a szakirodalomból származó irányítatlan átfedő közösségkereső segítségével teszteltük, valamint tovább vizsgáltuk az alapkoncepciót irányított hálózatok esetében is.

A következő, negyedik fejezetben egy olyan módszer került bemutatásra amely hálózatokban lejátszódó fertőzésalapú folyamatok minimalizálását teszi lehetővé célzott beavatkozások segítségével [120]. Az optimalizálási modell az Általánosított Független Kaszkádnak egy olyan kiterjesztése, ahol olyan adott elemű csúcshalmazt keresünk, melyek elemeinek fertőzésének a beavatkozással történő csökkentése maximalizálja a globális fertőzés csökkenését. A módszert egy pszichológiából származó

esettanulmány segítségével mutatjuk be, ahol a cél a negatív pszichológiai hatás csökkentése intézmények illetve cégek esetén, irányított pszichológiai beavatkozások segítségével. A teszteléshez valós adatot használtunk, valamint pszichológiatudományból származó szempontokat is figyelembe vettünk a modell kialakításakor.

Az ötödik fejezet a hálózatok céges illetve banki világban történő alkalmazását hivatott bemutatni, egy temporális banki utalási hálózatokon definiált új csalásdetektáló algoritmus segítségével [80]. A módszer egy a banki világban fontos utalási minta kinyerését teszi lehetővé melynek szereplői szorosan kapcsolódhatnak különböző csalásokhoz. Az algoritmust valós, 2016-ból származó banki anonimizált adatokon teszteltük, kitérve az utalási minta speciális eseteinek tárgyalására is. A módszer egy magyar bank csalásdetektáló rendszerének élesben használt része 2018 óta.

Az hatodik fejezetben a műszakkiosztási feladat megoldására mutattunk be egy olyan hálózatalapú módszert, ahol a feladat keresési tere gráfként van ábrázolva [81]. A feladat egy cég alkalmazottainak optimális foglalkoztatottságát próbálja megoldani úgy, hogy az megfeleljen különböző törvényi előírásoknak és költségében is optimális legyen. A módszer tesztelése valós és mesterségesen generált példákon történt, valamint a kapott eredményeket összehasonlítottuk az egészértékű programozási feladat által adott eredményekkel kis példák esetén.

7.2.1 Közösségkeresés és fertőzések modellezése tömegközlekedési hálózatokon

A bevezető után a disszertáció második fejezetét egy tömegközlekedési hálózatokban használható módszertan bemutatásával kezdtük amely alkalmas az utazási minták detektálására, valamint a hálózat fertőzési szempontból történő stressz tesztelésére [78] . Az első részben a módszer könnyebb megértése érdekében a tömegközlekedési együttutazási hálózat létrehozásához szükséges FAST-TrIPs travel demand modellt [100] valamint a hálózat struktúráját mutattuk be. Ezután definiáltuk a módszerünket melyet a következő négy lényegi lépésre lehet felosztani:

- A tömegközlekedési rendszer hálózatának a járatonként történő szétválasztása.
- Maximális klikkek detektálása a Bron-Kerbosch algoritmus segítségével [24].
- Transzfer hálózat F létrehozása.
- Közösség hálózat H létrehozása.

A módszertan a működése során a kezdeti contact hálózatból két további hálózatot épít fel. A contact hálózat az utasok közötti együttutazásokkal kapcsolatos információt tartalmazza. Az első (transfer hálózat) az utascsoportok közötti mozgást hivatott ábrázolni így lehetőséget adva a városban a sűrű járatkombinációk detektálására valamint a csoportok együtt mozgásának a követésére. A második létrehozott hálózat (community hálózat) pedig a csoportmozgások követése által felfedezett

utasok közötti rejtett kapcsolatokat tartalmazza. Az így létrehozott reprezentáció alkalmas arra, hogy szorosan összetartozó utasok közösségeit detektáljuk.

Járványterjedési alkalmazás

A módszer által adott finom rejtett kapcsolati struktúra lehetőséget ad arra, hogy a hálózaton történő vírusterjedés modelljét a járatok szintjén vizsgáljuk. A fejezet utolsó részében különböző utasokból indított fertőzési szimulációkat vizsgáltunk az SI modell segítségével. A módszerünk kimenete a tömegközlekedési járatok vírusterjedési szempontból történő kiértékelése és sorbarendezése. A fejezet végén a módszer Twin Cities Minneapolis tömegközlekedési hálózatán került kiértékelésre.

7.2.2 Fertőzésterjedés és közösségek

A harmadik fejezetben a fertőzésterjedés és az átfedő közösségi struktúra kapcsolatát vizsgáltuk tovább általános hálózatokban. A fertőzésmaximalizálás és a független kaszkád alapmodelljeinek bemutatása után úgynevezett redukciós módszereket vezettünk be, melyek alkalmasak, a fertőzésmaximalizálás keresési terének közösségkeresési módszerekkel történő csökkentésére. A hálózat csúcsai egy általunk bevezetett közösségi érték szerint lettek sorbarendezve fertőzőképességük szerint, és az így kapott felső X százalék adta $G(V*)$ halmazzal történt a fertőzésoptimalizálás. A közösségi érték alapötlete olyan csúcsok keresése, melyek több közösségben is megjelennek, biztosítva így a lehető legtöbb sűrű részgráffal való kapcsolatot a fertőzés terjesztésének érdekében. A módszer lépései a következők:

- Átfedő közösségkereső módszer futtatása a hálózaton.
- A közösségi értékek kiszámítása.
- Keresési tér szűkítése a kiszámított közösségi értékek alapján.
- Fertőzésmaximalizálás a mohó algoritmussal a szűkített keresési teret felhasználva.

Az általunk javasolt módszer alkalmas átfedő közösségkereső módszerek összehasonlítására is.

Fertőzésmaximalizálás és Irányítatlan Átfedő Közösségkereső Algoritmusok Összehasonlítása

A modell teszteléséhez nyolc különböző irányítatlan átfedő közösségkeresési módszer által adott eredményeket hasonlítottuk össze az eredeti mohó heurisztika eredményeivel [95] Lancichinetti és Fortunato [111] módszerével generált hálózatokon. Az eredmények alapján a ferőzésmaximalizálás szempontjából az InfoMap [155] közösségkereső adta a legjobb eredményeket. Általánosságban elmondható, hogy az általunk bemutatott közösségi értékalapú módszer alkalmas a fertőzésmaximalizálási feladat nagyobb hálózatokon történő megoldására is.

Irányított hub perkoláció

Az irányítatlan módszerek tesztelése után bemutattuk egy a szakirodalomban megtalálható irányítatlan közösségkereső algoritmus [28] irányított verzióját. A célunk az volt, hogy tovább vizsgáljuk a különböző közösséghez köthető struktúrák szerepét a fertőzésmaximalizálás szempontjából. A fejezet során az alapötletet kiterjesztve bemutattuk a hub értéket, amely a csúcshoz tartozó irányított teljes részgráfok száma alapján próbálja meghatározni a csúcs értékét. Az irányított esetben random és valós hálózatokon is teszteltük a módszerünket. A teszteléssel kapcsolatban elmondható hogy a módszertan mindkét esetben hatékonynak bizonyult.

7.2.3 Uplift hálózati modell célzott beavatkozások optimalizálására

A negyedik fejezetben egy hálózatokon lejátszódó, negatív folyamatok csökkentésére használható Uplift hálózati modellt mutattunk be célzott beavatkozások optimalizálására. A módszer az Általánosított Független Kaszkádon [29] történő fertőzésmaximalizálás modelljét terjeszti ki úgy, hogy egy referenciaszimulációhoz vett eltérést maximalizálva csökkenti a hálózat globális fertőzöttségét. A csúcsok különböző kezdeti fertőzöttségi értéket kapnak aszerint, hogy történt-e közbeavatkozás az adott entitás esetén vagy sem. A $G(V, E)$ hálózat csúcsai így $v^{apriori}$ valamint v^{uplift} valószínűségekkel rendelkeznek. A mohó algoritmus egy olyan halmazt keres amelynek csúcsainak a beavatkozáskor történő fertőzéscsökkenése, tehát $v^{apriori}$ valószínűségről v^{uplift} valószínűségre történő változtatása minimalizálja a globális fertőzöttséget.

Pszichológiai esettanulmány

A fejezet második részében a módszer egy pszichológiai alkalmazását mutattuk be ahol a cél negatív pszichológiai hatások csökkentése volt szociális hálózaton. A teszteléshez 14 darab, norvég szanatóriumból származó nővérek hálózatát vizsgáltuk és a WHO-5 pszichológiai kérdőívét [172] használtuk a pszichológiai állapotuk felmérésére. Az eredmények azt mutatták, hogy dolgozókkal történő célzott pszichológiai interakciók javíthatják a globális mentális állapotot a véletlen módon történő beavatkozásokkal szemben.

7.2.4 Időalapú hálózatok vizsgálata csalásdetektálásra a bankszektorban

Az ötödik fejezetben először ismertettük a csalásdetektálás alappilléreit, valamint definiáltuk a módszerünkhöz szükségek alapmodellt, majd bemutattuk a temporális hálózatokban utalási körök detektálására alkalmas algoritmusunkat. A temporális körökkel kapcsolatban elvárás, hogy a kör mentén az éleken lévő időbélyegeket időben egymás után következzenek, valamint a kör első utalási összegétől való eltérés nem lehet nagyobb egy bizonyos megadott α értéknél. A módszer a következő lépésekből áll:

- $H(P, E)$ hipergráf előállítása, vagyis a számlák azonos ügyfelek mentén történő csoportosítása.
- Idő szerint rendezett éllista előállítása.
- Módosított mélységi bejárás a rendezett éllista szerint.
- Paramétereknek megfelelő utalási körök kigyűjtése.

Az algoritmus bemutatása során az utalási körök speciális eseteit is vizsgáltuk, melyekkel kapcsolatban láthattuk, hogy a struktúra sajátosságai lényegesen eltérnek az általános gráfokkal kapcsolatos kördefinícióktól.

Esettanulmány banki tranzakciós hálózaton

A módszert egy magyar bank valós hálózatán vizsgáltuk, és az algoritmus mind futásidő tekintetében mind pedig a kívánt minták detektálása szempontjából hatékonynak bizonyult. A bemutatott algoritmus továbbá 2018 óta része az említett magyar bank csalásdetektáló rendszerének, és a működése során számos csalást sikeresen derített fel.

7.2.5 Hálózatalapú műszakkiosztás

Az utolsó fejezetben az ütemezési és műszakkiosztási feladatok témakörének rövid áttekintése után egy műszakkiosztási feladat megoldására alkalmas két lépéses gráfszínezési algoritmust vezettünk be. Az általános problémaleírás és a feladat matematikai modelljének a definiálását követően a heurisztikus módszerünk lépéseit mutattuk be amelyek a következők:

1. Kezdeti műszakkiosztás.

a. Dolgozók számának becslése.

b. Szabadnapminták generálása.

c. Konfliktus gráf felépítése.

d. A hálózat kezdeti színezése.

2. A hálózat újraszínezése Tabu Search algoritmus segítségével.

A módszert teszteltük véletlen generált és valós bemeneteken, valamint az általa adott eredményeket összehasonlítottuk az egészértékű programozási modell által adott eredményekkel kis gráfok esetén. Általánosságban elmondható, hogy a heurisztikus módszerünk az esetek nagy részében elérte az elméleti alsó korlátot és futásidő tekintetében is kielégítő eredményeket produkált.

7.2.6 További kutatási irányok

A disszertáció témaköreinek továbbfejlesztése számos lehetőséget rejt magában. A legkézenfekvőbb a módszerek különböző alkalmazási területekhez történő adaptálása, azonban a tárgyalt módszertanok továbbfejlesztése is kiterjesztése is nagy potenciállal bír.

Az első fejezetben bemutatott modellel kapcsolatban érdemes lehet egy olyan tömegközlekedési hálózatokon használható optimalizáló módszert fejleszteni, amely a járatok ütemezését úgy végzi, hogy figyelembe veszi az adott ütemezéshez kapcsolódó fertőzésterjedési szempontokat. Egy lehetséges továbbfejlesztési irány még a meglévő tömegközlekedési hálózatok újratervezése a vírusterjedés és az utasok kényelmének figyelembevételével.

A közösségstruktúra és fertőzési modellekkel kapcsolatban úgy érezzük, hogy a két terület a disszertációban tárgyalt módon történő összekapcsolása mindkét terület számára előnyökkel bírhat a jövőre nézve. A sűrű részgráfok és a fertőzésterjedés természetes kapcsolata sok helyen kihasználható olyan modellek és módszerek kifejlesztésére amelyek a közösségekhez köthető lokális minták felhasználásával globális hatást érhetnek el.

A harmadik fejezetben bemutatott módszer használható különböző való életben lejátszodó negatív folyamatok hatásainak a csökkentésére. A jövőben tervezzük a módszer epidemiológiai és pénzügyi területeken történő alkalmazását és kiterjesztését. A meglévő fertőzésmaximalizálással kapcsolatos tudás használata globális hatások csökkentése érdekében számos előnyt és potenciált tartogat magában a jövőre nézve.

A negyedik fejezettel kapcsolatban a jövőbeni terveink között szerepel egy olyan modell bevezetése ahol a különböző típusú gráfmotívumok és minták tetszőlegesen definiálhatók a módszer számára. Az így kapott keretrendszer egy olyan általános módszert adhat, amely képes bizonyos keretek között a felhasználó által definiált speciális lokális minták és struktúrák detektálására.

Az utolsó fejezetben bemutatott gráf alapú optimalizáló módszer a munkaerőkiosztás területén szolgálhat flexibilis valamint általános alapmodellként. A definiált modell olyan feladatreprezentációt használ amely lehetőséget adhat a jövőben a módszer más területekre történő kiterjesztésére, és további korlátozó feltételek kezelésére.

Bibliography

[1] Cfinder. http://www.cfinder.org/.

[2] Copra. https://gregory.org/research/networks/software/copra.html.

[3] Gce. https://sites.google.com/site/greedycliqueexpansion.

[4] Graph-tool. https://graph-tool.skewed.de/.

[5] Infomap. http://www.mapequation.org.

[6] Moses. https://sites.google.com/site/aaronmcdaid/downloads.

[7] Oslom. http://www.oslom.org/.

[8] Slpa. https://github.com/sebastianliu/SLPA-community-detection.

[9] J. Rousseau A. Wren. Bus driver scheduling: An overview. *Comput. Aided Sched. Pubilc Trahnsport.*, pages 173–187, 1995.

[10] E. Abbink, L. Albino, T Dollevoet, D Huisman, J. Roussado, and R. Saldanha. Solving large scale crew scheduling problems in practice. *Public Transport*, 3:149–164, 06 2011.

[11] A. Adamuthe and R. Bichkar. Tabu search for solving personnel scheduling problem. *Proceedings - 2012 International Conference on Communication, Information and Computing Technology, ICCICT 2012*, pages 1–6, 10 2012.

[12] C. Aicher, A. Jacobs, and A. Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3, 04 2014.

[13] W.J.K. Antoon, L.K. Jan, H.C. Papadimitriou, , and F. Spieksma. *Interval scheduling: A survey*, volume 54. Wiley Periodicals, Inc., 08 2007.

[14] D. Balcan, V. Colizza, B. Gonçalves, H. Hu, J. R. Ramasco, and A. Vespignani. From the cover: Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences of the United States of America*, 106:21484–9, 12 2009.

[15] J. Bao, C. Xu, P. Liu, and W. Wang. Exploring bikesharing travel patterns and trip purposes using smart card data and online point of interests. *Networks and Spatial Economics*, 17:1–23, 09 2017.

[16] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[17] S. G. Barsade. The ripple effect: Emotional contagion and its influence on group behavior. *Administrative Science Quarterly*, 47:644 – 675, 2002.

[18] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003.

[19] R. Bolton and D. Hand. Statistical fraud detection: A review. *Statistical Science*, 17, 08 2002.

[20] A. Bonifati and S. Dumbrava. Graph queries: From theory to practice. *ACM SIGMOD Record*, 47:5–16, 12 2018.

[21] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, April 1979.

[22] R. Briner and S. Reynolds. The costs, benefits, and limitations of organizational level stress interventions. *Journal of Organizational Behavior*, 20:647–664, 1999.

[23] D. Brockmann, Lars Hufnagel, and Theo Geisel. The scaling laws of human travel. *Nature*, 439:462–5, 02 2006.

[24] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973.

[25] A. Bóta, A. Csernenszky, L Győrffy, Gy. Kovács, M. Krész, and A. Pluhár. Applications of the inverse infection problem on bank transaction networks. *Central European Journal of Operations Research*, 23:345–356, 12 2014.

[26] A. Bóta and L. M. Gardner. Modeling the spread of infection in public transit networks: A decision-support tool for outbreak planning and control. *Transportation Research Board 96th Annual Meeting*, 01 2017.

[27] A. Bóta, L. M. Gardner, and A. Khani. Identifying critical components of a public transit system for outbreak control. *Networks and Spatial Economics*, 08 2017.

[28] A. Bóta and M. Krész. A high resolution clique-based overlapping community detection algorithm for small-world networks. *Informatica*, 39:177–187, 01 2015.

[29] A. Bóta, M. Krész, and A. Pluhár. Approximations of the generalized cascade model. *Acta Cybernetica*, 21:37–51, 01 2013.

[30] A. Carlsson-Kanyama and A-L. Lindén. Travel patterns and environmental effects now and in the future:: implications of differences in energy consumption among socio-economic groups. *Ecological Economics*, 30:405–417, 09 1999.

[31] C. Cattuto, M. Quaggiotto, A. Panisson, and A. Averbuch. Time-varying social networks in a graph database: [a neo4j use case]. *Neo4j*, 06 2013.

[32] M. Cepeda, R. Cominetti, and M. Florian. A frequency-based assignment model for congested transit networks with strict capacity constraints: Characterization and computation of equilibria. *Transportation Research*, 40:437–459, 12 2006.

[33] D.H. Chau and C. Faloutsos. *Fraud Detection Using Social Network Analysis, A Case Study*, pages 1–6. Encyclopedia of Social Network Analysis and Mining, 01 2016.

[34] M. Chen and H. Niu. A model for bus crew scheduling problem with multiple duty types. *Discrete Dynamics in Nature and Society*, 2012, 09 2012.

[35] N. Chen, L. M. Gardner, and D. Rey. Bilevel optimization model for the development of real-time strategies to minimize epidemic spreading risk in air traffic networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2569:62–69, 01 2016.

[36] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038, 09 2010.

[37] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 199-208:199–208, 01 2009.

[38] A. Chin, B. Xu, F. Yin, X. Wang, W. Wang, X. Fan, D. Hong, and Y. Wang. Using proximity and homophily to connect conference attendees in a mobile social network. *Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pages 79–87, 06 2012.

[39] B. Clarkson, C. Wagstaff, C. Arthur, and R. Thelwell. Leadership and the contagion of affective phenomena: A systematic review and mini meta-analysis. *European Journal of Social Psychology*, 50:61–80, 2020.

[40] R. Coenen and J. Broekens. Modeling emotional contagion based on experimental evidence for moderating factors. 2012.

[41] K. Danna and R. Griffin. Health and well-being in the workplace: A review and synthesis of the literature. *Journal of Management*, 25:357 – 384, 1999.

[42] G.B Dantzig. A comment on edies traffic delays at toll booths. *Operations Research*, 2:339–341, 1954.

[43] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborova. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 84:066106, 12 2011.

[44] J. Van den Bergh, J. Beliën, P. Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226:367–385, 05 2013.

[45] O. Diekmann and J.A.P. Heesterbeek. Mathematical epidemiology of infectious diseases: Model building, analysis and interpretation. *Wiley Series in Mathematical and Computational Biology, Chichester, Wiley*, 01 2000.

[46] P. Doreian, V. Batagelj, and A. Ferligoj. Generalized blockmodeling. *Cambridge University Press*, 25, 2005.

[47] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72:125–147, 1997.

[48] A. Downs, A. Boucher, A. G. Campbell, A. Laura A. Duncan, and G. A. Polyakov. Using the who–5 well-being index to identify college students at risk for mental health problems. *Journal of College Student Development*, 58:113 – 117, 2017.

[49] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 72:027104, 09 2005.

[50] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences of the United States of America*, 106:15274–8, 09 2009.

[51] D. Eisenberg, E. Golberstein, J. Whitlock, and M. F. Downs. Social contagion of mental health: evidence from college roommates. *Health economics*, 22 8:965–86, 2013.

[52] M. Elshafei and H. Alfares. A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs. *J. Scheduling*, 11:85–93, 04 2008.

[53] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. *Lect. Notes Comput. Sci.*, 6506, 06 2010.

[54] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[55] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[56] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: a review of applications, methods and models. *European Journal of Operational Research*, 153(1):3 – 27, 2004.

[57] A.V. Esquivel and M. Rosvall. Compression of flow can reveal overlapping-module organization in networks. *Physical review X*, 1(2):021025, 2011.

[58] I. Foppa. W.o. kermack and a.g. mckendrick: A seminal contribution to the mathematical theory of epidemics (1927). *A Historical Introduction to Mathematical Modeling of Infectious Diseases*, pages 59–87, 01 2017.

[59] S. Fortunato. Community detection in graphs. *Physics Reports*, 486, 06 2009.

[60] J. Fowler and N. Christakis. Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the framingham heart study. *The BMJ*, 337, 2008.

[61] B. Fredrickson. Cultivating positive emotions to optimize health and well-being. *Prevention & Treatment*, 3:1–25, 2000.

[62] B. Fredrickson. The role of positive emotions in positive psychology. the broaden-and-build theory of positive emotions. *The American psychologist*, 56 3:218–26, 2001.

[63] B. Fredrickson, Christian E. Waugh, and G. R. Larkin. Personality processes and individual differences what good are positive emotions in crises ? a prospective study of resilience and emotions following the terrorist attacks on the united states on september 11 th , 2001. 2003.

[64] A. Fukunaga, Hamilton E., Fama J., Andre D.O. Matan, and I.R. Nourbakhsh. Staff scheduling for inbound call and customer contact centers. *AI Magazine*, 23(4):30–40, 2002.

[65] S. Funk, M. Salathé, and V. Jansen. Modelling the influence of human behaviour on the spread of infectious diseases: A review. *Journal of the Royal Society, Interface / the Royal Society*, 7:1247–56, 09 2010.

[66] T. Gallai. Maximum-minimum sätze über graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 9:395–434, 1958.

[67] M. Gamache, A. Hertz, and J.O. Ouellet. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Comput. Oper. Res.*, 34(8):2384–2395, August 2007.

[68] L. M. Gardner, D. Fajardo, and S. Waller. Inferring infection-spreading links in an air traffic network. *Transportation Research Record: Journal of the Transportation Research Board*, 2300:13–21, 12 2012.

[69] M.R Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[70] M. Girvan and M. Newman. Community structure in social and biological networks. *proc natl acad sci*, 99:7821–7826, 11 2001.

[71] D. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[72] M.C. Golumbic. Algorithmic graph theory and perfect graphs. *Academic Press, San Diego, California,*, 1980.

[73] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[74] S. Gregory. Finding overlapping communities in networks by label propagation. *New journal of Physics*, 12(10):102018, 2010.

[75] R. Guimerà and L. Amaral. Functional cartography of complex metabolic networks. *Nature*, 23:22–231, 01 2005.

[76] L. Hajdu, A. Bóta, and M. Krész. Community based influence maximization in the independent cascade model. *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 237–243, 2018.

[77] L. Hajdu, A. Bóta, and M. Krész. Evaluating the role of community detection in improving influence maximization heuristics. *(Under Review)*, 2021.

[78] L. Hajdu, A. Bóta, M. Krész, A. Khani, and L. M. Gardner. Discovering the hidden community structure of public transportation networks. *Networks and Spatial Economics*, 20, 03 2020.

[79] L. Hajdu, B. Dávid, and M.s Krész. Gateway placement and traffic load simulation in sensor networks. *Pollack Periodica Pollack*, 16(1):102 – 108, 2021.

[80] L. Hajdu and M. Krész. Temporal network analytics for fraud detection in the banking sector. *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pages 145–157, 2020.

[81] L. Hajdu, A. Tóth, and M. Krész. Graph coloring based heuristic for crew rostering. *Acta Cybernetica*, (4):643–661, 2020.

[82] G. Hajos. Über eine art von graphen. *Int. Math Nachrichten*, 1957.

[83] E. Hatfield, J. Cacioppo, and R. Rapson. Emotional contagion by elaine hatfield. 1993.

[84] J. Heil, K. Hoffmann, and U. Buscher. Railway crew scheduling: Models, methods and applications. *European Journal of Operational Research*, 2019.

[85] A. Hesham. Survey, categorization, and comparison of recent tour scheduling literature. *Annals OR*, 127:145–175, 03 2004.

[86] A. Hevner, R. Alan, M. Salvatore, T. Salvatore, Park, P. Jinsoo, Ram, and Sudha. Design science in information systems research. *Management Information Systems Quarterly*, 28:75–, 03 2004.

[87] S. Hoogendoorn and P. Bovy. Pedestrian travel behavior modeling. *Networks and Spatial Economics*, 5:193–216, 01 2005.

[88] R. Huerta and L. Tsimring. Contact tracing and epidemics control in social networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 66:056115, 12 2002.

[89] J. Illenberger, K. Nagel, and G. Flötteröd. The role of spatial interaction in social networks. *Networks and Spatial Economics*, 08 2012.

[90] Cambridge Systematics Inc. 2010 travel behavior inventory: model estimation and validation report prepared for metropolitan council. 2015.

[91] III J. J. Bartholdi. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operations Research*, 29(3):501–510, 1981.

[92] D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4:77–84, 1975.

[93] M. Keeling and K. Eames. Networks and epidemic models. *Journal of the Royal Society, Interface / the Royal Society*, 2:295–307, 10 2005.

[94] V. Argilan A. Toth B. David C. Kemeny and G. Pongracz. Greedy heuristics for driver scheduling and rostering. *Proc. 2010 Mini-Conference Appl. Theor. Comput. Sci.*, pages 101–108, 2010.

[95] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137-146, 07 2003.

[96] B. Kernighan and Shou-De Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49:291–307, 1970.

[97] R. Kessler, P. Barker, L. Colpe, J. Epstein, J. Gfroerer, E. Hiripi, M. J. Howes, S. Normand, R. Manderscheid, E. Walters, and A. Zaslavsky. Screening for serious mental illness in the general population. *Archives of general psychiatry*, 60 2:184–9, 2003.

[98] A. Khani. Models and solution algorithms for transit and intermodal passenger assignment (development of fast-trips model). *PhD Dissertation, University of Arizona, Tucson AZ, USA*, 2013.

[99] A. Khani, T. Beduhn, J. Duthie, S. D. Boyles, and E. Jafari. A transit route choice model for application in dynamic transit assignment. *Innovations in Travel Modeling. Baltimore, MD*, 2014.

[100] A. Khani, M. Hickman, and H. Noh. Trip-based path algorithms using the transit network hierarchy. *Networks and Spatial Economics*, 15, 07 2014.

[101] D. König. *Theory of Finite and Infinite Graphs*, pages 45–421. Birkhäuser Boston, Boston, MA, 1990.

[102] L. Kovács. Conceptual systems and lexical networks in the mental lexicon (in hungarian: Fogalmi rendszerek és lexikai hálózatok a mentális lexikonban). *Tinta Könyvkiadó, Budapest*, 2013.

[103] L. Kovács, A. Bóta, L. Hajdu, and M. Krész. Brands, networks, communities: how brand names are wired in themind. *(Under Review)*, 2021.

[104] L. Kovács, A. Bóta, L. Hajdu, and M. Krész. Networks in the mind – what communities reveal about the structure of the lexicon. *Open Linguistics (To Appear)*, 2021.

[105] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborova, and P. Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences of the United States of America*, 110, 11 2013.

[106] M. Krész and A. Pluhár. *Economic Network Analysis Based on Infection Models*, pages 439–445. Encyclopedia of Social Network Analysis and Mining, Springer Science, 01 2014.

[107] R. Kumar and T. Calders. 2scent: an efficient algorithm for enumerating all simple temporal cycles. *Proceedings of the VLDB Endowment*, 11:1441–1453, 07 2018.

[108] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos. Edge weight prediction in weighted signed networks. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 221–230, 12 2016.

[109] W-T. Lai and C-F. Chen. Behavioral intentions of public transit passengers–the roles of service quality, perceived value, satisfaction and involvement. *Transport Policy*, 18:318–325, 03 2011.

[110] A. LaMontagne, T. Keegel, A. Louie, A. Ostry, and P. Landsbergis. A systematic review of the job-stress intervention evaluation literature, 1990–2005. *International Journal of Occupational and Environmental Health*, 13:268 – 280, 2007.

[111] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80:016118, 08 2009.

[112] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review. E*, 80(5):056117, 2009.

[113] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11, 03 2008.

[114] A. Lancichinetti, F. Radicchi, JJ. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[115] H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93–102, 1996.

[116] C. Lee, F. Reid, and A. McDaid abd N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *arXiv:1002.1827*, 2010.

[117] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2005.

[118] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans Knowledge Discov Data*, 1, 04 2006.

[119] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[120] D. Lipovac, L. Hajdu, S. Wie, and A. Q. Nyrud. Improving mental wellbeing in organizations with targeted psychosocial interventions. *Business Systems Research*, 11:86–98, 2020.

[121] A. Litan. The five layers of fraud prevention and using them to beat malware. *Gartner*, 2011.

[122] A. London, A. Háznagy, I. Fi, and T. Németh. Complex network analysis of public transportation networks: A comprehensive study. *International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 06 2015.

[123] L Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95 – 98, 1972.

[124] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253 – 267, 1972.

[125] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 01 2004.

[126] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1967.

[127] D. Marx. Graph coloring problems and their applications in scheduling. *Periodica Polytechnica, Electrical Engineering*, 48, 10 2003.

[128] A. McDaid and N. Hurley. Detecting highly overlapping communities with model-based overlapping seed expansion. *International Conference on Advances in Social Networks Analysis and Mining*, pages 112–119, 2010.

[129] A. Meisels and A. Schaerf. Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39(1):41–59, Sep 2003.

[130] M. Mesquita, M. Moz, A. Paias, and M. Pato. A decompose-and-fix heuristic based on multi-commodity flow models for driver rostering with days-off pattern. *European Journal of Operational Research*, 245(2):423 – 437, 2015.

[131] D. Montano, H. Hoven, and J. Siegrist. Effects of organisational-level interventions at work on employees' health: a systematic review. *BMC Public Health*, 14:135 – 135, 2014.

[132] Y-A. Montjoye, C. Hidalgo, M-Verleysen, and V. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 03 2013.

[133] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: An overview, ii. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 09 2017.

[134] N. Nassir, A. Khani, M. Hickman, and H. Noh. Algorithm for intermodal optimal multi-destination tour with dynamic travel times. *Transportation Research Record: Journal of the Transportation Research Board*, 2283:57–66, 12 2012.

[135] M Newman. Spectral methods for community detection and graph partitioning. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 88:042822, 10 2013.

[136] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69:026113, 03 2004.

[137] E.W.T. Ngai, H. Yong, Y.H. Wong, Y. Chen, and X. Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50:559–569, 02 2011.

[138] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics Theory and Experiment*, 2009, 02 2008.

[139] K. Nurmi, J. Kyngäs, and G. Post. Driver rostering for bus transit companies. *Engineering Letters*, 19(2):125–132, December 2011.

[140] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 07 2005.

[141] G. Palla, I. Farkas, P. Pollner, I. Derenyi, and T. Vicsek. Directed network modules. *New Journal of Physics*, 9, 04 2007.

[142] S. Pandit, D.H. Chau, S-Y. Wang, and C. Faloutsos. Netprobe: A fast and scalable system for fraud detection in online auction networks. *16th International World Wide Web Conference, WWW2007*, 01 2007.

[143] A. Paranjape, A. Benson, and J. Leskovec. Motifs in temporal networks. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 02 2017.

[144] K. Peffers, T. Tuunanen, C. Genglers, M. Rossi, W. Hui, V. Virtanen, and J. Bragge. The design science research process: A model for producing and presenting information systems research. *Proceedings of First International Conference on Design Science Research in Information Systems and Technology DESRIST*, 02 2006.

[145] T. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 89:012804, 01 2014.

[146] TP. Peixoto. Model selection and hypothesis testing for large-scale network models with overlapping groups. *Cambridge University Press*, 5(1):011033, 2015.

[147] R. Pendyala, K. Konduri, Y-C. Chiu, M. Hickman, H. Noh, P. Waddell, L. Wang, D. You, and B. Gardner. An integrated land use–transport model system with dynamic time-dependent activity-travel microsimulation. *91st Annual Meeting of the Transportation Research Board, Washington, DC*, 2303, 12 2012.

[148] X. Qiu, W. Cen, Z. Qian, Y. Peng, Y. Zhang, X. Lin, and J. Zhou. Real-time constrained cycle detection in large dynamic graphs. *Proc. VLDB Endow.*, 11(12):1876–1888, August 2018.

[149] U.N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[150] G. Ramadurai and S. Ukkusuri. Dynamic user equilibrium model for combined activity-travel choices using activity-travel supernetwork representation. *Networks and Spatial Economics*, 10:273–292, 06 2010.

[151] D. Rey, L. Gardner, and S. Waller. Finding outbreak trees in networks with limited information. *Networks and Spatial Economics*, 16:687–721, 2016.

[152] K. Richardson and H. Rothstein. Effects of occupational stress management intervention programs: a meta-analysis. *Journal of occupational health psychology*, 13 1:69–93, 2008.

[153] R. Rocha and B. Thatte. Distributed cycle detection in large-scale sparse graphs. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 08 2015.

[154] J. N. Rosenquist, J. Fowler, and N. Christakis. Social network determinants of depression. *Molecular Psychiatry*, 16:273–281, 2011.

[155] M. Rosvall and C. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105:1118–1123, 02 2008.

[156] M. Saberi, T. Rashidi, M. Ghasri, and E. Kenneth. A complex network methodology for travel demand model evaluation and validation. *Networks and Spatial Economics*, 04 2018.

[157] G.G. Sadowksi and P.Rathle. Fraud detection: Discovering connections with graph databases. *Neo4j White Paper*, 2015.

[158] M. Salathé, M. Kazandjieva, J-W. Lee, P. Levis, M. Feldman, and J. Jones. A high-resolution human contact network for infectious disease transmission. *Proceedings of the National Academy of Sciences of the United States of America*, 107:22020–5, 12 2010.

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

[160] Y. Shen and R.S.K Kwan. *Tabu Search for Driver Scheduling*, pages 121–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[161] C. Song, Z. Qu, N. Blumm, and A.L. Barabasi. Limits of predictability in human mobility. *Science (New York, N.Y.)*, 327:1018–21, 02 2010.

[162] A. Srivastava, C. Chelmis, and V. Prasanna. The unified model of social influence and its application in influence maximization. *Social Network Analysis and Mining*, 5, 12 2015.

[163] L. Sun, K. Axhausen, D-H. Lee, and M. Cebrian. Efficient detection of contagious outbreaks in massive metropolitan encounter networks. *Scientific Reports*, 4, 05 2014.

[164] L. Sun, K. Axhausen, D-H. Lee, and X. Huang. Understanding metropolitan patterns of daily encounters. *Proceedings of the National Academy of Sciences*, 01 2013.

[165] S. Szabó and B. Zaválnij. Coloring the nodes of a directed graph. *Acta Universitatis Sapientiae. Informatica*, 6:117–131, 04 2014.

[166] R. Tarjan. Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 114–121, 1971.

[167] A. Tucker. Chapter 2: covering circuits and graph colorings. *Applied Combinatorics (6th Edition) 49.*, 2006.

[168] J. V. D. van der Klink, R. Blonk, A. Schene, and F. V. van Dijk. The benefits of interventions for work-related stress. *American journal of public health*, 91 2:270–6, 2001.

[169] R. Wang, F. Nakamura, T. Okamura, and H. Warita. How the risk-taking personality influences commute drivers' departure time choices. *Procedia - Social and Behavioral Sciences*, 16:814–819, 12 2011.

[170] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[171] J. West and M. Bhattacharya. Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, 57:47 – 66, 2016.

[172] WHO. Wellbeing measures in primary health care/the depcare project. 2020.

[173] Z. Wu, Y-F. Lin, S. Gregory, H-Y. Wan, and S-F. Tian. Balanced multi-label propagation for overlapping community detection in social networks. *Journal of Computer Science and Technology*, 27, 05 2012.

[174] Liu X Xie J, Szymanski BK. Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. *IEEE 11th international conference on data mining workshops*, pages 344–349, 2011.

[175] H. Yin, A. Benson, J. Leskovec, and D. Gleich. Local higher-order graph clustering. *KDD : proceedings. International Conference on Knowledge Discovery & Data Mining*, 2017:555–564, 08 2017.

[176] Y. Yuan and G. Gay. Homophily of network ties and bonding and bridging social capital in computer-mediated distributed teams. *J. Comput. Mediat. Commun.*, 11:1062–1084, 2006.

[177] B. Zaválnij and S. Szabó. Coloring the edges of a directed graph. *Indian Journal of Pure and Applied Mathematics*, 45:pp 239–260, 04 2014.

[178] S. Zhang, R-S. Wang, and X. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications*, 374:483–490, 01 2007.