Resource Allocation in Networks

Alvis Logins

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

Resource Allocation in Networks

A Dissertation
Presented to the Faculty of Science and Technology
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Alvis Logins
February 17, 2020

Abstract

Given the spatial locations of customers and a road network, where to build new facilities that would satisfy the customers needs? How to allocate vaccination centres in a country to suppress a virus epidemics? Which Twitter accounts can advertise a piece of news to the largest auditory and in the most robust way? Prima facie, the questions are disparate, but all derive from the generic problem of *Resource Allocation in Networks*. Given network nodes representing a set of consumers and a set of possible resource locations, the goal is to minimize the loss or maximize the profit of allocating a limited budget of indivisible resources. Interacting with a resource occurs via a connection from supply to demand nodes through network edges with certain characteristics like capacity, cost, weight, or probability. Such a process may occur with or without a restriction of flow preservation at nodes, by which inflow to a node is equal to outflow, resulting to a *transportation* in the former case and to a *diffusion* in the latter case.

This thesis investigates methods and solutions for this family of resource allocation problems with respect to the transportation and diffusion models, building on previous work in the fields of Operations Research, Machine Learning, and Data Management, considering temporal and stochastic components. We propose novel techniques to calculate model parameters and to achieve objectives of convenience in facility location, fairness in vehicle cruising, and robustness in diffusion control problems, with a good balance between scalability and quality. Our experimental studies based on real-world datasets show the efficiency and effectiveness of the solutions.

Resumé

Givet kunders geografiske placering og et vejnet, hvor skal man så bygge nye faciliteter, som kan tilfredsstille kundernes behov? Hvordan identificerer man de bedste placeringer af vaccinationscentre for at stoppe en virusepidemi? Hvilke Twitter-konti kan annoncere en nyheder til den største målgruppe på den mest robuste måde? Ved første øjekast er spørgsmålene forskellige, men alle stammer fra det samme generiske problem - ressourcetildeling i netværk. Givet netværksknuder, der repræsenterer et sæt forbrugere og mulige ressourceplaceringer, er målet at minimere tabet eller maksimere overskuddet ved at allokere en begrænset mængde af udelelige ressourcer. Interaktion med en ressource sker via en forbindelse fra udbuds- til efterspørgselsnoder gennem netværkskanter, som har egenskaber så som kapacitet, omkostninger, vægt eller sandsynligheded. En sådan proces kan forekomme med eller uden en begrænsning af strømningskonservering ved knudepunkter, hvor indstrømning til en knude er lig med udstrømning, hvilket resulterer i en transport i førstnævnte tilfælde og en diffusion i sidstnævnte tilfælde.

Denne afhandling undersøger metoder og løsninger til gruppen af ressourcefordelingsproblemer med hensyn til transport- og diffusionsmodellerne, baseret på tidsmæssige og stokastiske egenskaber. Afhandlingen bygger på tidligere arbejde inden for områderne operationsanalyse, maskinlæring og dataforvaltning. Vi foreslår nye teknikker til beregning af modelparametre og til at opnå målsætninger om bekvemmelighed i facilitetsplacering, retfærdighed i køretøjscruising og robusthed i diffusionskontrolproblemer med en god balance mellem skalerbarhed og kvalitet. Vores eksperimentelle studier, som er baseret på virkelige datasæt, viser ydeevnen og effektiviteten af løsningen.

Contents

Part I

Overview

Chapter 1

Introduction

During World War II an excessive amount of research groups were formed by the British government in order to find better analytical methods for the utilization of military resources. That period is arguably considered the beginning of the *Operational Research* (OR) field [245]. OR is a branch of applied mathematics that overlaps with mathematical optimization and statistical analysis. The general model in OR can be formulated as a constrained maximization of profit or minimization of loss. Investment, Production planning, Urban Development planning are just a few examples where OR is an indispensable tool. Main methods include Linear Programming, Integer Programming, Dynamic Programming, Network Programming, Nonlinear Programming, and such heuristics as Simulated Annealing, Genetic Algorithms, and Local Search [245]. *Resource Allocation in Networks* (RAN) is one of the central class of problems in OR, appearing as the Facility Location problem [84], Budget Allocation problem [239], Online Assignment problem [39], Firefighter problem [10], and others [245], depending on the type of resources and constraints in a network.

Data Science (DS) has opened new horizons in Resource Management, and Predictive Analytics in general. It mixes Statistics, Machine Learning, Database Management, and Distributed Systems [9], to deal with large scale datasets, and attack real-world problems too complex for traditional mathematical modeling and existing OR computational methods. Such tools as Clustering, Data Indexing, Association Rules, Data Visualization, and Machine Learning with Neural Networks in particular, are typically associated with modern DS [78]. A problem may appear in DS literature under a unique name, although being partially addressed in OR. Such examples are the Optimal Location Query [57] problem, a special case of Facility Location problem, and the Skyline Query [38] problem, known in OR as Pareto-efficient Resource Allocation [141] problem. Nevertheless, there are differences in methodologies and scalability of the solutions, designating DS as the *Data-Intensive* field [116].

Resource Allocation is a broad topic that deserves a field by itself. We investigate a class of optimization problems of allocating limited indivisible resources at *nodes of a network*. In the following chapters, we summarize existing solutions in the fields of OR and DS, show their advantages and limitations. With an emphasis on scalability, we propose novel algorithms based on practical heuristics and data management techniques.

The thesis is divided into two parts. In the first part, we give an overview of our research and a high-level description of the field. The second part consists of our publications and extensions. Each chapter of the second part corresponds to one or two publications and a preface with general information on the content. Topicwise, our study is organized based on the presence of *temporal* and *stochastic* components in a model, and interaction principles between participating agents (*consumers*) and the resource.

We distinguish two types of interaction, namely a *flow-type* and a *diffusion-type*. Models with the flow-type interaction are also known as *transportation network* models. Consumers obtain a resource though a *flow* in a network. The amount of flow coming into a node must be equal to the amount of flow coming out of the node, for all nodes except the locations of consumers and resources. This interaction type models transportation systems, supply chains, traffic in computer networks, currents in a power grid. In contrast, *diffusion-type* interaction does not impose the flow preservation restriction. For instance, a source of information in a social network, serving as a resource, can reach each subscriber (consumer), independently on the number of subscribers. Likewise, an infected person can spread a disease to any number of fellows, independently on how many people could have had infected this person.

We start with a *static* and *deterministic* RAN with the flow-type interaction. The model does not have a temporal component and does not deal with uncertainty. Chapter 2 provides an introductory RAN problem of this category, the Bipartite Matching problem, followed by Chapter 3 with some basic tools used in RAN. Chapter 4 introduces the Facility Location problem with an emphasis on the static deterministic case, and also briefly discuss dynamic and stochastic versions of the problem. Facility Location is a flow-type RAN that relates to road networks and involves spatial data management. Chapter 8 presents a new scalable heuristic for the static deterministic Facility Location problem.

Then, we move to the dynamic *and* stochastic RAN with the flow-type interaction, where a model supports decisions over time under uncertainty. Chapters 5 and 9 present our investigation of this type of problems. We focus on the models that follow the Markov Decision Process framework, where an optimization algorithm has to *train* a model that is able to *predict* future profitability of an allocation decision. We are particularly interested in how *fair* is the trained model with respect to consumers. In the second part of the

thesis, we propose our adaptations of existing techniques that boost fairness in the context of taxi fleet management.

Further, Chapter 6 gives an overview of RAN with the diffusion-type interaction, also known as *diffusion network* models. Most of the problems of this interaction type are both dynamic and stochastic, although we mention some exceptions. We focus on RAN for *information spread control* in social networks, although our study can be applied for other diffusion-type phenomena, e.g. epidemic prevention. An allocated resource can be either a source of a spread (a *seed*), e.g. a community blog, or a *barrier* for the spread. An example of a barrier allocation is the *Node Immunization* problem, where a limited amount of *vaccines* need to be allocated in a geosocial network to suppress a virus. Another example is defending nodes of a computer network to prevent a spread of malware. In Chapter 10 we first conduct an experimental study where we compare the state-of-the-art solutions to the node immunization problem. Then, we describe a novel the framework for training a real-world network model for information spread, which we also used in the first part of the chapter.

Finally, we study a game-like setting, when the allocation of barriers for a spread is combined with the allocation of seeds. The quality of seeding is evaluated by its *robustness* against immunization. Chapter 7 shows the connection between the robustness, fairness, and Game Theory. Chapter 11 presents our research on robustness of diffusion. We propose new measures of robustness, evaluation algorithms, and heuristics to improve the robustness.

The research is primarily conducted by the author of the thesis. The ideas are the result of a joint effort of the student, the main supervisor and the external collaborators. Coding, methodology, experimental study, text of the introduction are authored by the student. The text of publications is written by the student and edited by the main supervisor.

Chapter 2

Bipartite Matching

Given a set of workers and tasks, represented as a bipartite graph, the objective of the *Bipartite Matching problem* is to find matches between workers and tasks such that the cost of the matches is minimized. In this chapter, we provide a brief historical overview of this introductory RAN problem, starting with a special case of the problem, the balanced Assignment problem, where the number of workers is equal to the number of jobs and each worker can be assigned to any job. Then, we introduce concepts of alternating paths and matching augmentation. Next, we introduce the problem of flow cost minimization, show its connection to the Bipartite Matching problem, define a residual graph, the Capacity-Scaling and the Cost-Scaling methods. Generalizations of the Bipartite Matching problem to the domains of dynamic and stochastic models are included in the following chapters. In Chapter 8 of the second part, we use the introduced definitions, concepts and methods to derive a novel RAN algorithm.

2.1 Assignment Problem

In 1931 Dénes König, a Hungarian mathematician and author of the first book on graph theory [179], proved the König's theorem, that describes an equivalence between the maximum matching and the minimum vertex cover problems in bipartite graphs. In the same year, another Hungarian mathematician Jenö Egerváry independently proved the equivalence in a more general case of weighted graphs. In 1955 Harold Kuhn published the famous *Hungarian algorithm* [149], deriving the theoretical basis of the algorithm from the works of König and Egerváry.

The Hungarian algorithm solves *the Assignment Problem*. We outline the graph formulation of the algorithm. Let $G = (V, E)$ be a complete bipartite graph. A set of vertices is divided into two subsets $V = S \cup D$, representing workers and jobs (Figure 21a). The task is to find *a perfect matching*, a subset of edges $E' \in E$, such that each vertex is incident exactly to one edge in E'.

During the execution of the algorithm, each vertex maintains *a potential* (a label) $p(v_i) \to \mathbb{R}$, that satisfies two invariants

$$p(v_i) - p(v_j) \leq c(v_i, v_j)$$

$$p(v_i) > 0$$

where $(v_i, v_j) \in E$, and $c(v_i, v_j)$ is *a cost* (a weight) assigned to the corresponding edge. The algorithm starts with $p(v_i) = 0$, $\forall i$, and with *a direction* of edges from S to D. At each step, the algorithm does a search of existing paths from S to D along edges where potentials of incident nodes are *tight*:

$$p(v_i) - p(v_j) = c(v_i, v_j)$$

Such paths are called *alternating*, since edges along a path alternate the direction in respect to two node sets of the bipartite graph. An alternating path that begins and ends with a free (not-matched) node is called *an augmenting path*. For each path found, the algorithm reverse edges along the path. The reversing operation is called *matching augmentation*. When there are no such paths left, the potential of the reachable nodes in S is increased by

$$\Delta p = \min_{i,j}\{c(v_i, v_j) - p(v_i) - p(v_j) \,|\, v_i \in S, v_j \in D\}$$

The potential of the reachable nodes in D is decreased by the same value. A set of edges with orientation from D to S represents the matching. The algorithm terminates when the matching is perfect.

The Hungarian algorithm stands first in the line of *combinatorial optimization* algorithms that use *the duality principle*. The principle says that an optimization problem can be viewed from a perspective of the primal or the dual problem. The difference between the solutions to the primal and the dual problems is called *the duality gap*. If the gap is zero, then the optimal solutions are equal between problems (the strong duality theorem). Furthermore, any feasible solution to the dual problem is an upper bound to any solution of the primal problem (the weak duality theorem). In the case of the Hungarian algorithm, finding the potentials is a dual problem to the matching problem. The duality theorems were first conjectured by John von Neumann in 1947, and are one of the central theorems in mathematical optimization theory.

2.2 From Assignments to Flows

In 1956, Ford and Fulkerson extended the Hungarian algorithm to the problem of computing the maximum flow in a network [86]. Given a source node, a sink node, and edge *capacities*, the Ford-Fulkerson algorithm applies the same concept of reversing utilized edges as the Hungarian algorithm. At each iteration, the algorithm finds the maximal flow between the source and the

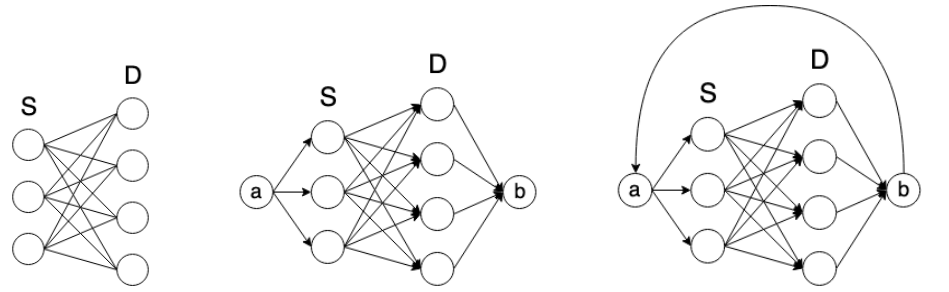
sink. The *flow* is a path together with the maximal edge capacity along that path. Then, capacities of the edges along the path are decreased, while the capacity of the *virtual* edges in the opposite direction is increased, as an analogy of reversing edges in the case of the Hungarian algorithm. The graph with changed capacities and virtual edges is called *a residual graph*. Once there is no any flow from the source to the sink in the residual graph, the algorithm terminates.

The Blossom algorithm by Edmonds [80] was inspired by the Ford-Fulkerson algorithm [75], and released almost a decade after its publication. It constructs the maximum matching in *a general graph*. The idea is to iteratively find augmenting paths, perform matching augmentation, and contract the alternating paths that create *cycles*. One such cycle is called a *blossom*, which then can be represented by a single vertex in further algorithm iterations. The algorithm has polynomial complexity, and motivated Edmond to propose polynomial time as a measure of goodness of algorithms in Computer Science [86]. Furthermore, as a part of the analysis of the algorithm, Edmonds created a technique of describing a convex hull of matching solutions by linear inequalities, that allowed to prove various theorems in matching theory as special cases of the Duality theorem [179]. This resulted in a separate branch of Combinatorial Mathematics – Polyhedral Combinatorics [179].

The Ford-Fulkerson and Blossom algorithms appeared as a cornerstone in the line of optimization algorithms in networks. Later, the problem was generalized to *the Minimum Cost Flow Problem*, and further to *the Minimum Cost Circulation Problem* (MCCP). Edge *weights* were introduced, representing the cost of sending a unit of flow along an edge. The goal of the problems is to find a flow with the minimum cost. In MCCP, instead of source and sink constraints, flow requirements are defined through lower bound labels on edges. MCCP can be reduced to the most general formulation of the assignment problem, where matches are weighted, and there may be several matches, or none, per worker or per job. We refer to this generalization of the assignment problem as *the Bipartite Matching problem*.

The reduction from MCCP to the bipartite matching problem is illustrated in Figure 21. Figure 21a shows the initial complete bipartite graph, with workers S and jobs D. Assume edges are labeled by weight (a cost of assignment of a worker to a job), and capacity (number of times a worker can be assigned to a job). Similarly to the Hungarian algorithm, we add directions to edges from S to D. Then, we add two extra nodes a and b, representing a source and a sink (Figure 21b). a is connected to each node in S, and the capacity of edges between a and S is set to the total maximum number of jobs a particular worker can handle (let us call it a worker capacity). Finally, let the total required flow from source to sink be equal to the sum of worker capacities. For simplicity, assume that the sum is equal to the number of jobs. Finding a flow from source to sink of a minimal cost will correspond to the matching of the minimal cost, given that one unit of flow from S to

D represents one match. This completes the reduction to the Minimum Cost Flow problem. Further, let us add an arc from b to a, and set the lower bound for the flow along that arc to the required flow from source to sink (Figure 21c). Then, a and b become regular nodes with the flow preservation, and the problem reduces to MCCP. Note, that instead of a complete bipartite graph we can also restrict matchings to a set of feasible matches. In case the sum of worker capacity is not equal to the number of jobs, we should add an extra job (or extra worker), and allow to be matched with it for an infinitely large cost. The details of this case can be found in [257].



(a) Assignment (b) Minimum Cost Flow (c) Minimum Cost Circulation

Figure 21: The Bipartite Matching problem

Methods that use the concepts of path augmentation and a residual graph are known as *Capacity-Scaling* methods. One famous representative is the Edmonds-Karp algorithm [81], that finds the residual graph using Breadth-First Search, instead of Depth-First Search used by Ford and Fulkerson. Another seminal Capacity-Scaling method is the *Successive-Shortest Path Algorithm* (SSPA) by Derigs (1981) [75]. It computes the minimum weight perfect matching problem in a *general graph*. It uses the shortest paths for building a residual graph, and a labeling procedure that is a modification of the Blossom algorithm [75]. Although the idea of using the shortest paths in the implementation of the Hungarian algorithm existed since 1960th [46, 118], the advantage of SSPA is that it allows a simple and efficient implementation, and was first to use purely combinatorial arguments, without going into polyhedral combinatorics [75]. It makes the algorithm especially appealing to the DS community. A special case of SSPA for *bipartite graphs* uses a simpler *distance-based labeling technique* [5, 257]. Nodes are assigned potentials, similarly to the potentials used by the Hungarian algorithm, that indicate the proximity of the node to the end of the shortest alternating path passing the node. The labeling technique significantly reduces the time required for finding an augmenting path. In 2010, Leong Hou U et al. [257] observed that the efficiency of SSPA in this case can be significantly improved by pruning redundant edges in a bipartite graph, using an additional data structure that keeps incident edges in sorted order and a tight threshold that depends on

the potentials. We improve on this result and utilize the pruning technique to develop a scalable RAN algorithm, described in Chapter 8.

2.3 From Capacity-Scaling to Cost-Scaling

The idea of distance-based potentials was first presented by Goldberg and Tarjan [100] as a part of the *Push-Relabel algorithm* for computing maximum flows in a network [5]. Let an edge be *admissible* in respect to the potential function iff

$$p(v_i) - p(v_j) = c(v_i, v_j) + 1$$

A flow though the admissible edges is called a *preflow*. The algorithm iteratively increases the preflow toward the sink (the *push* operation), and increases potentials of nodes to a minimum value required for at least one admissible out-edge (the *relabel* operation). The algorithm terminates when there is no applicable operation to be done.

The concept of a preflow was invented earlier by a Soviet mathematician Karzanov [100]. The Push-Relabel algorithm is a result of applying the distance-based labeling to the algorithm of Karzanov. A generalization of the Push-Relabel algorithm is the *Cost-Scaling* methods for the Minimum Cost Flow problem. The idea of Cost-Scaling was originally proposed by Röck [216], and then used by Goldberg and Tarjan [101] in the *ϵ-scaling algorithm*. The algorithm relaxes the admissibility, allowing a preflow to follow paths which are *approximately* admissible, such that

$$p(v_i) - p(v_j) - c(v_i, v_j) \leq \epsilon$$

Then, ϵ is iteratively decreased in a geometric progression, followed by the pushing and the relabeling operations.

Cost-Scaling methods are considered ones of the most efficient flow optimization algorithms [6]. Furthermore, they can be efficiently parallelized and applied to the bipartite matching problem [170]. A distributed counterpart of the ϵ-scaling algorithm is called an *Auction algorithm*, presented by Bertsekas in 1979 [29]. On the negative side, Cost-Scaling methods can not benefit from sequential consideration of edges, as in SSPA [170], so the computational optimization methods of Leong Hou U et al. [257] do not apply.

Notably, there exists a combinatorial flow optimization algorithm that does not use any scaling, the *Cycle-Cancelling algorithm*, also proposed by Goldberg and Tarjan [100], as well as the *Double-Scaling algorithm* that uses both scaling methods simultaneously [4]. All algorithms are strongly polynomial. For a comprehensive summary of the bipartite matching problem and flow optimization algorithms, refer to [5, 46, 179].

2.4 Applications of Bipartite Matching

Solutions to the Bipartite Matching problem can be used in some problems even if there is no direct reduction to the Bipartite Matching problem [179]. Examples are the Chinese Postman problem (minimization of distance while traversing all streets, polynomially solvable), the Travelling Salesman problem (minimization of distance while visiting all intersections, np-hard), a minimum weight spanning tree problem, and variations of the Facility Location problem [46, 126, 224]. The WMA algorithm presented in Chapter 8 is another example. Generalizations of the Bipartite Matching problem, namely Online and Stochastic Bipartite Matching, appear in Machine Learning, Online/Adaptive Learning and are commonly considered in allocation problems over time under uncertainty. We discuss this generalizations in Chapters 4 and 5 of the thesis.

Chapter 3

Standard Tools

In the following sections, we introduce several standard optimization methods, used across other chapters of the thesis. *Linear Programming* (LP) is a well-known approach for optimizing a linear objective function. We use it as one of the baselines in Chapter 8, and discuss the applicability of LP in the rest of the RAN problems in the corresponding chapters. *Submodularity* is a property of an objective function that allows applying a wide range of approximation methods for maximization and minimization tasks, including a simple greedy node selection. This property is a basis for the state-of-the-art algorithms for diffusion control, which we study in chapters 10 and 11. Finally, we briefly discuss *Dynamic Programming* and define the Bellman equation, used in chapters 5, 6 and 9, and the *Monte-Carlo* method, used in chapters 5, 6, 10 and 11.

3.1 Linear Programming

Starting from the works of Leonid Kantorovich in 1930th and the duality principle by John von Neumann in 1947, *Linear Programming* has become the central tool for solving a wide range of optimization problems [179], and RAN in particular. The in-depth investigation of the general state-of-the-art LP-techniques is out of the scope of this thesis, so we provide only a brief introduction to the basic ideas. A discussion on the applicability of LP to the RAN problems studied in the thesis is included in the follow-up chapters. The content of this chapter is based on the book of Luenberger and Ye [182], which we recommend for further reading.

The *standard form* of a linear program is

$$\begin{aligned} &\min \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} > 0 \end{aligned} \tag{3.1}$$

where $\boldsymbol{c}$ and $\boldsymbol{A}$ is a vector and a matrix of coefficients, and $\boldsymbol{x}$ are the unknowns. A problem with a linear objective function and linear constraints

can be transformed into the standard form. The Bipartite Assignment problem and the Minimum Cost Circulation problem are special cases of a linear program.

Consider a feasible solution $\boldsymbol{x}$ of size n, s.t. $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is an $m \times n$ matrix. Let $\boldsymbol{B}$ be an $m \times m$ matrix build on any m independent columns of $\boldsymbol{A}$. Then, a solution where all elements of $\boldsymbol{x}$ that are not associated with $\boldsymbol{B}$ (non-basic variables) are set to zero is a *basic* solution with respect to the *basis* $\boldsymbol{B}$.

The fundamental theorem of LP, due to Caratheodory [180], states that if there is a feasible solution for LP, then there is a basic feasible solution; and if there an optimal feasible solution, then there is a basic optimal feasible solution. The theorem reduces a search of optimal solutions to a search of basic optimal solutions.

The Simplex Method is the first algorithm in LP by Dantzig (1951) [72], which finds optimal solutions though a process of *pivoting*. Consider a basis $\boldsymbol{B}$. By linear transformations and variable reordering, we can bring the equation $\boldsymbol{Ax} = \boldsymbol{b}$ to the *canonical form*:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{B} \end{bmatrix} \boldsymbol{x} = \boldsymbol{b'}$$

where $\boldsymbol{I}$ is a unity matrix. The array of coefficients $\boldsymbol{A'} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B} & \boldsymbol{b'} \end{bmatrix}$ is called a *tableau*. Let x_q be a non-basic variable and x_p be a basic variable of the reordered $\boldsymbol{x}$. The goal of the pivoting process is to make x_q basic and x_p non-basic. To do so, we take the *pivot element* a_{pq} of $\boldsymbol{A'}$, divide p-th row of $\boldsymbol{A'}$ by a_{pq} to get a unit coefficient for x_p in p-th equation, and then subtract multipliers of p-th row from other rows so that coefficients for x_q are zero in all other equations. That makes x_q basic, as a result of x_p becoming non-basic, yielding a new basis $\boldsymbol{B'}$. The pivot element is selected considering a *cost* of bringing a new element to the basis

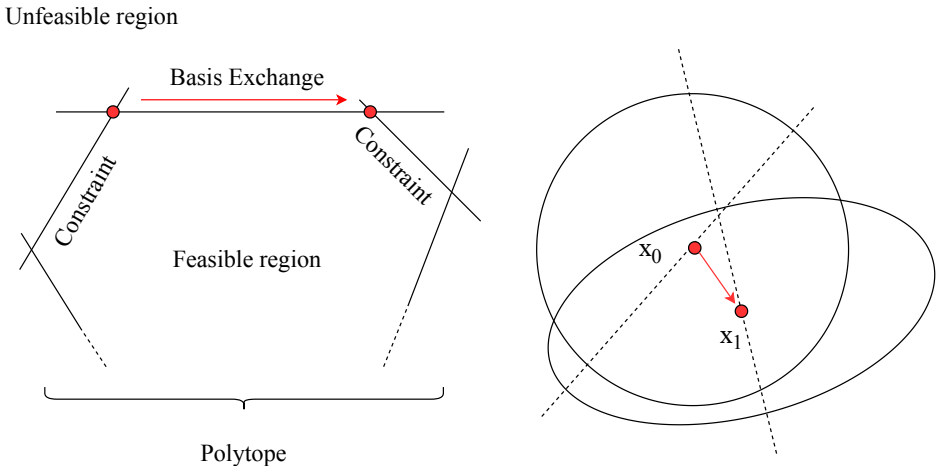
$$r_p = c_p - \sum_{i=1}^{m} c_p a_{ip}$$

where c_p is an element of the cost vector $\boldsymbol{c}$ in Eq. 3.1. The exchange of basis to be advantageous if the cost is negative. If there are no such exchange, then the current basic feasible solution is optimal, and is equal to the optimal feasible solution.

The geometrical interpretation of the Simplex method is illustrated in Figure 31a. Feasibility constraints define a polytope. Basic feasible solutions and their basis correspond to the vertices of the polytope, and the pivoting process corresponds to moving from one vertex to an adjacent one. The Simplex method is a textbook representative of a more general *basis-exchange* methods in LP. In contrast, the *interior-point* methods, firstly suggested by Neumann, is another class of optimization algorithms that traverse the *interior* of the feasible region until they reach the best solution.

One of the most important representatives of the interior-point methods is the *Ellipsoid method*. The optimization objective is defined by a binary search. For each objective candidate, the method checks if the candidate is feasible. An objective value defines a feasibility polytope, and the method checks if there is any feasible point within the polytope. If so, the objective value is feasible, and the point corresponds to the solution. The process of finding a feasible point is illustrated in Figure 31b. Starting with the feasible polytope P and the ellipsoid that is guaranteed to contain P, the method checks if the center of the ellipsoid x_0 is feasible. If yes, then the goal is reached. If not, the method finds a *cutting plane* (a dashed line) that goes thought x_0 with the side of the plane containing P. Based on it, the next ellipsoid with the center in x_1 is derived, which is guaranteed to have a *smaller volume*.

The Ellipsoid Method, developed in 1960th in the Soviet Union, was an important step for the problems with the exponential number of constraints, such as the maximum matching problem, which now can be solved just by a polyhedral description of the convex hull of matchings [179]. While the simplex algorithm is not polynomial, the Ellipsoid method is polynomially bounded, which makes it very useful in the optimization theory for complexity analysis. However, it is not faster than the Simplex algorithm in practice. The state-of-the-art LP optimization algorithms combine the interior-point methods with the duality theory and techniques of *convex optimization*.



(a) Simplex method (b) Ellipsoid method

Figure 31: LP optimization

One of the most important convex optimization technique is based on the use of the *Lagrange multipliers*, which are called the dual variables in LP. The general idea is to substitute the objective function in Eq. 3.1 with the Lagrange function defined by

$$L(\boldsymbol{x}, \lambda) = \boldsymbol{c^T x} - \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{Ax}) \tag{3.2}$$

where $\boldsymbol{\lambda}$ are Lagrange multipliers. In case of a single constraint, the optimal solution to the original problem is one of the *stationary points* of L, where all first partial derivatives of L are zero. Therefore, finding an optimal solution boils down to finding stationary points. For several constraints the method extends with similar arguments.

Lagrangian relaxation is a relaxation method in OR, that use the Lagrange multipliers to simplify an optimization problem. The simplified optimization objective is the maximization of the unconstrained $L(x, \lambda)$, where the violation of the constraints is penalized using the Lagrange multipliers, instead of satisfying them strictly. λ represents the weights for the penalization.

In RAN, variables are restricted to be integers. It is a special case of LP, which is NP-complete and generally harder then the continuous counterpart. Most of the OR algorithms for the flow-type RAN are based on Lagrangian relaxation [258]. Other optimization techniques such as Stochastic Gradient Descent [39], cutting-plane methods, and various heuristics like local search or genetic algorithms also apply.

3.2 Submodularity

Submodularity is a generalization of convexity to set functions. A function $f(S)$ is *submodular* if and only if the marginal gain from adding an element to S is decreasing with the size of S (a property of *diminishing returns*). There are 3 equivalent definitions:

- $\forall A \subseteq B,\ f(A \cup \{s\}) - f(A) \leq f(B \cup \{s\}) - f(B)$
- $\forall A,\ B,\ f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$
- $\forall A \subseteq B,\ s \notin B,\ f(A \cup \{s\}) - f(A) \geq f(B \cup \{s\}) - f(B)$

Minimum of an unconstrained submodular objective is computable in polynomial time. Minimization under constraints, as well as any submodular maximization is an NP-hard problem. However, the beautiful property of the maximization problems with submodular objective is that a *greedy heuristic* for such problems yield an approximation guarantee of $(1 - 1/e)$ ($\sim 63\%$), or $1/2$ if the problem is unconstrained, which is also the best possible guarantee unless P=NP [147]. This property is applied for large-scale FL problems [84, 142], stochastic spread control in diffusion networks [138], as well as in a wide range of machine learning tasks, such as feature selection, active learning, ranking [148]. We utilize this property to build new algorithms in Chapter 11.

Unlike submodular maximization, the approximation guarantee does not hold for the minimization problem, and there are no existing efficient solutions for the minimization problem in a general case [128]. Nevertheless, the field of minimizing unconstrained submodular functions is rapidly developing in

recent years due to its applicability in Machine Learning (ML) [77, 128], and RAN [92]. Jegelka and Krause [128] provide a comprehensive overview of the state-of-the-art applications of the submodular optimization in ML, including minimization, maximization, and active learning. Gamlath et al. [92] apply LP, the constructive version of the Caratheodory's theorem [180], and the submodular minimization methods to solve the *Stochastic Bipartite Matching* (SBM) problem, which is a generalization of the Bipartite Matching problem to the case when *edges* are unknown. A solution to SBM should provide edges sequentially in an order desirable for matching. An edge proposed by the solution represents a *query* (an offer) to a customer. The *query-commit* SBM requires that the first offer accepted by a customer must appear in the final matching. The SBM *with Price-Of-Information* allows several offers to the same customers independently on the acceptance, but each offer indues a cost. SBM *with a few queries* is another variation where the cost is zero, but the number of queries is limited. Recently, Yamaguchi et al. [275] proposed a proof technique for the performance analysis of a generic SBM, including the mentioned ones and a generalization of the bipartite matching to hypergraphs.

One of the most scalable approximation approaches to the submodular minimization is the *Minimum Norm Point algorithm* (MNP) [90, 128]. MNP is a gradient descent algorithm that builds on the Lovász extension of the submodular objective. The *Lovász extension* is a reduction from the minimization of a submodular *set* function to the constrained minimization of a convex *continuous* function. The number of constraints is exponential in number of variables, but due to the Ellipsoid method, described in the previous section, the minimum can be computed in polynomial time. Recently, Chakrabarty et al. [48] suggested to improve the gradient descent by providing a fast calls to an objective oracle, that leads to a subquadratic runtime of the MNP algorithm. Their solution is based on an additional data structure, an enhanced binary search tree, that allows to query the decision variables while updating them. This improvement is of a high importance, because the Ellipsoid method used in achieving the prior theoretical results is numerically unstable and have a poor practical performance.

A special case of submodular minimization is the *Graph Cut* problem [77, 128]. A scalable approach to the minimization problem, widely used in the computer vision field, is based on the idea of approximating the objective by the second-order graph cut objective. The cut problem might be linearly solvable in most cases. The recent paper of Djolonga et al. [77] shows the high interest of the AI community in the topic. Authors study the behaviour of the MNP algorithm in application to Deep Neural Networks. Jegelka et al. built on the idea of utilizing the graph cuts and suggested a generalized submodular graph cut problem to approximate a broader class submodular functions, including the constrained submodular minimization [48, 129]. Other more recent works on special cases of submodular minimization include [125, 191]. Results of Jegelka et al. were later applied by Staib and Jegelka to the

Budget Allocation problem [239], a special case of RAN that mixes flow-type and diffusion-type interactions. We cover it in more details in Chapter 7.

3.3 Dynamic Programming

Dynamic Programming (DP) is a method for optimal control developed by Richard Bellman in 1950s. The core idea is to *divide-and-conquer*, i.e., split a larger problem into smaller sub-problems, and solve them recursively. The recursive connection between the problem and its sub-problems is defined through a *Bellman equation*. Consider a dynamic decision problem with infinite time horizon as an example. Let x_t be a state of a system at the time moment t. Let a_t be a decision on some control variable that leads to a change of the state. Let $V(x_t)$ be a *value function* that shows a future profitability of the state, and $R(a_t, x_t)$ be a pay-off from making the decision. Then, the problem is to maximize $V(x_0)$, and the corresponding Bellman equation that recursively defines the function is

$$V(x_t) = \max_{a_t}\{R(a_t, x_t) + \gamma V(x_{t+1})\} \tag{3.3}$$

where x_{t+1} is the new state of the system, and γ is a *discount factor*, a parameter that is used for convergence results of certain algorithms, and represents the probability that the infinite dynamic process will terminate.

In Chapter 6 we provide Bellman Equations for toy examples of RAN, the Facility Location and the Influence Maximization problems, that illustrate the connection between RAN of the flow-type and the diffusion-type interactions. Larger and more complex problem instances are barely handled by the classic DP approach, due to its poor scalability. Additionally, DP requires a perfect and complete model, which often is intractable for complex stochastic environments with a temporal component due to so called "curse of dimensionality".

More advanced methods such as Approximate DP [210], Adaptive DP [99], and Neuro-Dynamic Programming [30] are suitable for RAN, and improve the scalability of the DP approach by function approximations, but still are limited to hundreds of decision variables, which does not suffice for the real-world problems considered in this thesis. We leave the detailed review of these methods out of the scope of the thesis.

3.4 Monte-Carlo

A *Monte-Carlo* (MC) method is a generic term for all optimization methods that use a suitable amount of randomly generated observations to obtain a numerical estimation of a solution [270]. Likewise, an MC *simulation* is a generic term for methods that estimate an objective by sufficient random sampling. It is a common approach in problems with significant uncertainty and

high degrees of freedom, widely applied in the fields of engineering, biology, economics, and computer science.

In this thesis we use MC in Chapters 9, 10 and 11. Algorithms that use MC in connection to the diffusion control (Chapters 10, 11) are also referred as Sketching algorithms [165]. As a general idea, they sample network *scenarios* to collect experience, and then allocate resources at nodes that appeared as the most influential, or, in reverse, nodes that fall under influence more often. In Reinforcement Learning (Chapter 9), MC is used together with DP in what is called the Time-Difference (TD) method. Unlike MC, TD methods partially utilize experience from other learned estimates. For other applications of the MC method and MC simulations, refer to the book Rubinstein and Kroese [223].

Chapter 4

Facility Location

The general formulation of the *Facility Location* problem (FL) is to allocate a set of facilities (resources) so that to maximize the satisfaction of a demand (customers), or minimize their inconvenience [84]. A distinctive characteristic of FL among other resource allocation problems is the *flow-type* consumer-resource interaction and *the spatial nature* of the problem. Customers and facilities are presented as objects in a continuous 2-dimensional space, or attributed to road network elements. Therefore, various techniques like advanced routing algorithms, clustering, spatial indexing, and pruning based on spatial location apply [188].

In this chapter, we propose a classification of Facility Location problems based on the type of facilities and whether customers are *points* or *trajectories*. We also discuss the *Routing problem* (Section 4.5), in the context of its application to FL. A separate branch of research is dedicated to the effect of *dynamic prices* on the allocation of facilities in case of the probabilistic demand-supply. We cover the routing and pricing problems in more detail due to the increasing interest in the topics from the DS community, and the connection to our overview of submodularity applications (Section 3.2) and online recommendations (Chapter 5). In Chapter 8 of the second part of the thesis, we study the static deterministic version of FL, where both customers and facilities are nodes of a road network.

4.1 Classification of Facility Location Problems

Table 41 summarizes the classification of FL, grouped by the type of customers and facilities. For a summary of methods applied to the discrete FL problems, refer to [84, 258]. The most studied variation of the Facility Location problem is the one where both customers and facilities are represented as static *points*. If points are nodes in a road network, then the problem becomes a set cover problem. Facilities can be *capacitated*, meaning that each facility can serve a limited amount of customers. *Hub points* are a special case of static point

facilities, usually representing public transportation stations. A hub is a node such that it must be visited by a customer to get to the destination. *Competitive Facilities* are points or trajectories, which belong to a different "players on the market". This means there is more than one objective function, each corresponds to one "player". Facilities are placed in iterations, maximizing the corresponding function. *Obnoxious facilities* are points or trajectories which negatively influence customers, so that the objective function usually maximizes a dispersion of facilities. *Temporal* facilities and customers are those who have defined time-dependent working hours or demand accordingly. We cover the case of customers as trajectories in Section 4.5.

Customers / Facilities	Points	Stochastic Points	Temporal Points	s-d Trajectories	Stochastic Trajectories	Given Trajectories
Uncapacitated Points	[26, 27, 57, 60]	[85, 237]	[203]			[91, 188]
Capacitated Points	[11]			[296]		
Hub Points	[218]				[164]	
Competitive Facilities	[1, 87]			[273]		
Obnoxious Facilities	[28]		[40]	[8]		
Temporal Facilities			[130, 184]		[183]	
Built Trajectories	[121, 134]		[154, 220, 280]			
Given Trajectories	[54, 276]					

Table 41: FL Summary

Most of FL problems can be formulated as an LP program. Hence, various LP techniques apply, like Mixed Integer programming [130, 188], Binary Integer programming [188, 193], Lagrangian relaxation. The majority of approximation algorithms use a rounding technique for the generalized assignment problem [235] as a basis.

4.2 Classic Facility Location

In this thesis we focus on a canonical case of FL, namely the vertex k-median problem, where we minimize the distance between nodes and the *closest* facility by selecting k nodes for new facilities. The LP program for the k-center problem is shown in Eq. 4.1. Variable x_i is an indicator that a_i is selected for a new facility. Variable y_{ij} is another Boolean that captures whether node a_i is assigned to facility a_j. A set of constraints in Eq. 4.2 assures that a_i can be assigned only if a_j has a facility and the total amount of new facilities is k. Due to Eq. 4.2 and the monotonically increasing $d(i, j)$, the optimal assignment will always correspond to the assignment to the closest facility.

$$\min_{y_{ij}} \sum_i \sum_j d_{ij} y_{ij}, \quad x_j, y_{ij} \in \{0, 1\} \tag{4.1}$$

$$y_{ij} \leq x_j, \quad \sum_j x_j = k, \quad \sum_j y_{ij} = 1 \tag{4.2}$$

Although LP and convex optimization provides an accurate solutions with a great flexibility in constraints and objective, such accurate methods aim datasets of thousands of nodes and hundreds of variables. Approximation methods based on LP increase these numbers on an order of magnitude [40]. At the same time, road maps of large cities as Copenhagen or Moscow have over billion nodes and edges, and some social networks from Standford Large Network Dataset Collection [156] have over 10^7 edges. For such real-world datasets LP can not provide a solution in a reasonable time, so the challenge is to provide faster algorithms keeping the objective as close to the optimal as possible. In Chapter 8 we present a novel heuristic-based Wide Matching Algorithm (WMA) that solves the capacitated k-center problem for large datasets, the variation of the problem where each facility can serve only several customers up to some capacity value. WMA uses a repeated matching as the main action of the algorithm. While the first idea solving discrete FL problem by repeated matching was introduced in 1999 by Rönnqvist et al [224, 258], they use LP to evaluate the "score" of the current matching, and, if infeasible, they use a randomized heuristic to "split" some facilities and redistribute customers. We provide a more scalable solution, utilizing a greedy set cover routine instead of LP when evaluating matching feasibility, as well as consider a more general problem formulation.

4.3 Scalable Facility Location

Scalable solutions for FL appear in the literature as Optimal Location Queries. In this problem a facility can locate at any point on an edge. Yao et al. [280] suggested a divide-and-conquer algorithm that partitions the network into subgraphs, using a clustering algorithm. Chen et al. [57] proposed a solution which takes advantage of a precalculated neighborhood, the *Nearest Location Component* (NLC). NLC contains all points that are closer to that customer than the nearest already placed facility. The solution ranks such intervals by the number of customers that have this interval in their NLC. Other computational optimization techniques are spatial indexing techniques (kd-trees and R-trees [136]), and space division. We use the space division techniques in Chapters 8 and 9 of our thesis, and give more examples in Section 4.6.

4.4 Stochastic Facility Location

Stochastic Facility Location is a class of FL problems where customers, resources or network elements are probabilistic. One example of such model is the k-median problem with stochastic edge length and demands described by discrete *scenarios*. The problem can be solved by Lagrangian Relaxation [236], treating the stochastic FL as a deterministic FL of a larger scale. Complexity of the problem in this case grows linearly with the number of scenarios. Richer models may include probabilistic production costs, selling prices, probabilistic disruption of facilities, emergency needs of customers, and dynamic reallocation with extra cost. For a complete survey, refer to Snyder et al [236].

4.5 Route Planning

Route Planning is an important component in FL algorithms. A basic routing in a deterministic network implies finding shortest paths and is usually implemented by the *Dijkstra's algorithm* [76], also used in Chapter 8 of this thesis. Let H be a min-priority queue, and the goal is to find a shortest path length between v and w. The algorithm starts by initiating H with a single node v. Then, iteratively, the algorithm pulls a node u from H, and updates distances to all neighbours of u to

$$\text{dist}(n) = \min\{\text{dist}(n), \text{dist}(u) + \text{weight}(u, n)\}$$

where n is a neighbour node, $\text{dist}(n)$ is a current known distance from v to n, and $\text{weight}(u, n)$ is a weight (cost) of an edge between u and n. Distances are initiated with an infinitely large number, and the algorithm terminates when w is pulled from H. At termination, $\text{dist}(w)$ stores the shortest path length. In order to find the shortest path, the algorithm saves path predecessors in an additional array each time the distance is updated. The path is obtained by *tracing back* once the algorithm reaches w. If applied on road networks, a simple efficiency improvement can be achieved by maintaining two heaps instead of one, and running two instances of the algorithm from v and w simultaneously. The stopping criterion in this case is when the sum of the top values of two queues is larger or equal to the length of the best path seen so far. For the details of the *Bidirectional* Dijkstra's algorithm and other basic Point-to-Point Shortest Path algorithms, refer to Goldberg et al [102]. Notably, Bidirectional Dijkstra's algorithm is a special case of a general bidirectional search heuristic, that is faster than an unidirectional counterpart only for specific domains [22].

Scalable routing algorithms use the domain-specific knowledge. For example, road networks are planar networks, with uniform degree distribution, and uniform branching factor, that makes the bidirectional search heuristic effective. Other than that, the networks have a highly *hierarchical structure*. This

fact is used in the *Transit Node Routing* approach. The idea is to precompute distances between small set of transit nodes interconnected by a sparse network relevant for long-distance travel. Caching and spatial indexes are also widely used in route planning.

Among the vast amount of computational optimization methods for finding the shortest paths [201], a ground-breaking data-intensive technique was proposed by Geisberger et al [95], called *Contraction Hierarchies*. It also exploits the hierarchical structure of road networks by building a set of *shortcuts*. The algorithm has a preprocessing phase, and a query phase. At the preprocessing phase, nodes are iteratively removed from a network, and if a removed node is a part of the shortest path between its neighbours, a new virtual edge (a shortcut) is added. At the query phase the shortest path is computed by the Bidirectional Dijkstra's algorithm, but due to the shortcuts, the algorithm can skip unimportant vertices, i.e. such vertices that do not lie on the shortest path. During the tracing back phase of the Dijkstra's algorithm, the shortcuts are *unpacked* to obtain the shortest path. Finding an optimal sequence of nodes in the preprocessing phase in an NP-hard problem, so various heuristics apply. Overall, the technique allows to query the shortest path in a road network with over 10^7 nodes within a fraction of second [95]. Contraction Hierarchies are used in modern car-navigation systems, route planners, traffic simulators [96].

Best routes in real-world applications do not always follow the shortest paths. Traffic conditions, speed limit, and even amount of right turns along a route can influence preferences of drivers and arriving time. In OR this case is covered by assigning a certain probability distribution to edges. Then, for example, for each path the expected arrival time is equal to the convolution of distributions of all edges along the path. However, such approach leads to a large accumulative error, has limited scalability due to the costly convolution operator, and does not account for such factors as turns and personal preferences. Furthermore, trajectory datasets might be too sparse to collect enough information about each edge. An innovative approach was proposed by Yang et al. [71]. They proposed to use trajectories directly instead of calculating cost distribution per-edge, so to preserve implicit dependencies between edges. Then, a path cost is "collected" by finding the largest available segments in the trajectory data. Numerous works follow this trajectory-oriented paradigm, including works on specialized indexing techniques [268] and stochastic Contraction Hierarchies [207].

In relation to FL, trajectory-oriented approach appears when trajectories represent *customers*, and some works are mentioned in Table 41. *s-d trajectories* are the type of customers that allow detours, and hence are represented only by source and destination. *Stochastic trajectories* imply some assumptions on the probabilistic distribution of customers and their trajectories. *Given trajectories* corresponds to the case described above – an algorithm processes (indexed) set of historical trajectories. Funke et al. [91] is an im-

portant representative of the last category. Authors formulate the *Hitting Set problem* – to find a minimal set of nodes that covers all customer trajectories. Customers are assumed to travel along shortest paths. It is a generalization of the k-path cover problem, when the paths should not be obligatory shortest, and which, in its turn, is a generalization of a set cover problem. The solution is based on Contraction Hierarchies.

In Chapter 8 we present the WMA algorithm for capacitated FL, with customers represented as nodes. It is based on successive solutions of the bipartite matching problem. Throughout the algorithm, we maintain two instances of the Dijkstra's algorithm, for solving the shortest path problem in a bipartite graph (following the approach of the SSPA algorithm), and the shortest path in a road network. WMA iteratively builds the bipartite network, until there exist a set cover of a specific size for all the customers. We describe the details in Chapter 8, but we note the connection of our approach to the routing planning algorithms, covered in this section. Firstly, more efficient shortest path solutions can be embedded in our algorithm for better performance. Secondly, WMA can be adapted to customers represented by trajectories, by redefining the shortest distance to a customer as the shortest distance to any point of his trajectory. Similar idea of finding a set cover over customers-trajectories was used by Funke et al. [91] in the Hitting Set problem. We leave the extension of WMA to the trajectory-based demand for future work.

4.6 Dynamic Pricing

Dynamic Pricing in RAN can significantly influence allocation decisions. In case of flexible prices, models involve price-sensitive functions of supply and demand, so maximization of a particular objective, like revenue, throughput or social welfare, also implicates finding an equilibrium price. The term *dynamic* additionally suggests time dependency of the functions. Dynamic Pricing is most useful when the product (resource) has an expiration date and its capacity is fixed [187]. Although we do not study pricing in the following chapters, we describe few works on the topic in this section as an introduction to taxi management systems (Chapter 9), competitiveness and fairness (Chapter 11).

Pricing strategies have been excessively studied in OR [61, 208], and FL in particular [84, 236]. Overviews of Dynamic Pricing models in OR are presented by Bitran et al. [32] and Elmaghraby et al. [83]. Two outstanding case studies of Dynamic Pricing application are the airline seat pricing [187], and the Surge Pricing model of a ride-hailing company Uber. The pricing strategy of airlines is one of the most complex in economics theory. Development of the Dynamic Pricing model is often credited to American Airlines Robert Crandall, as a response to the success of a competitor in early 1980s. In 1994, the company was estimated to produce extra $500 million revenue per year

based on its Dynamic Pricing management techniques [187]. More recently, Uber has proved another remarkable success of Dynamic Pricing [108]. In 2012, Uber's Boston team noticed that on Friday and Saturday drivers tend to go home early, leaving partygoers alone and unhappy. As a solution, Uber suggested a higher price during that time period, and two-thirds of the unfulfilled requests were eliminated. The story has attracted attention of the DS community [252], and together with the rising popularity of ride-hailing systems, an actively developing research branch emerged [53]. Other industries where the Dynamic Pricing approach has found its way are hotels, car rentals, subscription systems and advertisement [61].

A representative work on data-intensive dynamic pricing in ride-hailing systems was published by Tong et al [252]. Given a spatial region (a city) partitioned into *grid cells*, and a set of *workers* (drivers) to complete a set of *tasks* (customers), the goal is to maximize the expected revenue from completing tasks. Time horizon is divided into *discrete time steps*, and for each time step the problem is solved independently (no *prediction*). A driver can serve a customer within certain radius. A customer is willing to accept a driver with a probability that depends on the price. If accepted, the driver relocates to the destination according to the customer request. The *Global Dynamic Pricing* problem is to determine prices per each cell, and *dispatch* drivers, i.e. assign drivers to customers.

The proposed solution to the Global Dynamic Pricing problem is referred as Matching-based Prising Strategy (MAPS). It is based on ideas from theory of *auctions* [16] and submodularity of a proxy objective. Authors note that for a single sell the optimal price is located on the intersection of *expected* demand and supply curves, the demand curve is known to be convex, and the supply curve is a straight line. Figure 41 illustrates the concept. As a cell receives larger expected supply, the supply line raises towards the revenue axis. Formally, if l is a cell id, n_l is an expected number of drivers *assigned* to customers in the cell, p_l is a price, and $S_l(p_l)$ is an acceptance ratio of customers, then a revenue of the cell is equal to

$$L_l = \min\{p_l \cdot S_l(p_l), n_l \cdot p_l\}$$

In the formula, we assume that all customers travel the same distance for simplicity. Each price value corresponds to a particular optimal expected car distribution across cells n_l. Therefore, for each car distribution there is a price value it corresponds to. The task of the algorithm becomes to find an optimal n_l for each l, under the constraint of current car locations.

The extension of their solution to the Facility Location is trivial. Let cars be facilities with capacity of one, and without current location. The goal is to find optimal locations for cars. One can simply increase n_l in a greedy manner, selecting l where L_l benefits the most. In MAPS driver assignments are *constrained* by distances, so the optimality of the greedy approach is not

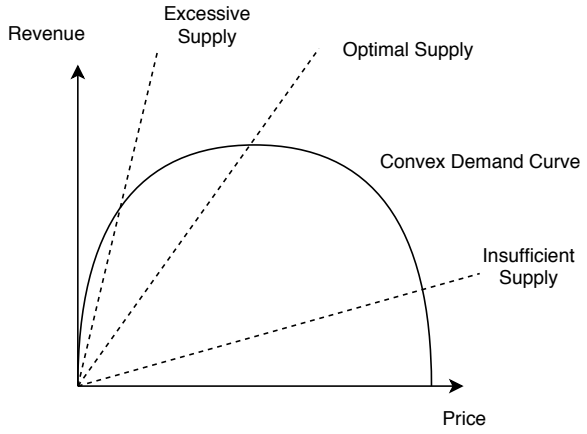


Figure 41: Optimal pricing under uncertain demand

trivial. Tong et al. prove that the objective of maximizing $\sum_l L_l$ under spatial constraints is submodular, so indeed greedy approximation apply. Defining n_l for each l is *the preliminary driver assignment*. On top of that, MAPS considers the demand curve unknown, and explores iteratively with time, using the Upper Bound Algorithm (UCB), inspired from Babaioff et al [16]. We cover UCB in details in Section 5. Once the price is set, customers make a decision whether to accept the price. This results in a deterministic bipartite graph of possible matchings between drivers and customers. As the final stage of MAPS, drivers are dispatched by solving the bipartite matching problem.

MAPS represents a common approach to *simulate* a taxi fleet [49, 53, 133, 167, 252, 295]. Cars are relocating in a grid in discrete time steps, serving customers. Cars within same cell are indistinguishable, and customers are attached to cells rather than specific locations. Cars make a joint decision, implying that the overall system architecture is centralized. We use similar approach in Chapter 9. In our work, we consider a problem of *cruising*, i.e. relocating idle drivers who don't get an order. The Global Dynamic Pricing problem is a special case of *the dispatching problem*, which is complimentary to the cruising problem. Unlike MAPS, we predict distributions of customers and relocation of drivers, that eliminates such effects as *wild goose chasing* [47], i.e. picking up distant customers and wasting drivers time and earnings.

MAPS is based on Myerson auction mechanism [16]. The mechanism is known to violate fairness among participants [255], unlike, e.g., the oldest but not widely used Vickery mechanism [13]. Recently, Zheng et al. addressed this issue by proposing social-welfare oriented auction mechanism in the same context of order dispatching [294]. In our work, we also address the fairness issue, but from driver's perspective, orienting our solution to the poorest driver. We talk about the fairness and the worst case oriented solutions in Chapter

11 in details. For an overview of various market models refer to Reynolds et al [214].

Other important works on Dynamic Pricing in the taxi management context include Balseiro et al [18], where authors consider a model of sequential querying of customers, and several works [14, 53] where solutions follow the Markov Decision Process framework, discussed in the next chapter.

Chapter 5

Decisions Over Time Under Uncertainty

Sometimes resource allocation decisions should be provided or updated on-the-fly, responding to a live stream of resource availability information and timely prediction of user preferences. For example, a ride-hailing company should maintain a distribution of cars in a road network in a way that matches stochastic demand patterns. Failing to do so would lead to a drop in the revenue of the company, unequal workload among drivers, and unsatisfied customers. A model for such problems involves both a *temporal* component (e.g., dynamics of cars, time dependency of the demand), and a *stochastic* component (e.g., orders appear in random). Models with the temporal component are also referred as *dynamic*, or a *control* process.

A common approach is to model a stochastic control process is *Markov Decision Process* (MDP). One or several *actors* perform one of possible *actions*. An *environment* produce a *reward* for the performed action, and defines (deterministically or with uncertainty) its next *state*. The objective is to maximize a sum of rewards over *an episode*, a sequence of *discrete time periods*. Actors follow *a policy*, which is a deterministic or probabilistic function that maps a state to action. An action depends only on the current state, which is a core property of MDP. Three fundamental classes of methods for solving finite Markov decision problems are Dynamic Programming, Monte-Carlo methods, and Temporal-Difference (TD) learning. TD learning is a core method in the field of Reinforcement Learning (RL), which is the focus of this chapter.

5.1 The Multi-Armed Bandit

The Multi-Armed Bandit (MAB) problem aims to allocate limited resources when parameters of a model are only partially known to an algorithm. The name comes from gambling at slot machines, known as one-armed bandits. It is a classic problem where a solution should involve an *exploration-exploitation*

balance, a decision on when to explore new solutions, and when to follow an existing strategy. The problem is often used as an introductory problem to RL since it illustrates the concept of *evaluative* feedback of training data. The feedback provides an instant *value* for any choice, in contrast to instructive feedback on whether a certain choice is a *correct* choice.

Following the MDP framework, the classic MAB assumes that a reward follows a stationary unknown probability distribution, $p(r|a)$, where a is the taken action and r is a reward for that action. At each time step, an agent selects an action according to the current knowledge of the distribution (a policy), and after receiving feedback from the environment (a reward), the agent can adjust the knowledge according to the new experience.

Let A be a set of possible actions, and $\pi_t(a)$ be the policy – a probabilistic distribution over possible actions at time moment t. Let $Q(a)$ be a *value* function that estimates an expected reward from an action a. Given the rewards up to the time moment t, we can estimate a future reward as an average over experienced rewards:

$$Q_t(a) = \frac{\sum_{i=1}^{t} r_i[a_i = a]}{\sum_{i=1}^{t}[a_i = a]}$$

where $[a_i = a]$ is an indicator that is equal to 1 if an action at time moment i was the same as the evaluated action a, otherwise 0. Then, $Q(a) = \lim_{t\to\inf} Q_t(a)$. Let A_t be a set of actions that are the best according to the experience, for example actions with the largest average revenue so far:

$$A_t = \arg\max_{a \in A} Q_t(a)$$

Then, a greedy policy for the Multi-Armed Bandit problem is

$$\pi_t(a) = \frac{[a \in A_t]}{|A_t|}$$

The drawback of the greedy solution is that there might not be enough statistics for some actions. Therefore, an algorithm should take into consideration the *exploration-exploitation* compromise. Instead of always selecting the greedy action, it can randomly select one of the available actions. A simple implementation is the *ϵ-greedy strategy*:

$$\pi_t(a) = (1 - \epsilon)\frac{[a \in A_t]}{|A_t|} + \frac{\epsilon}{|A|}$$

where ϵ is a parameter that shows how often an algorithm prefers the greedy strategy over the random policy. A common practice is to decrease exploration ϵ with time.

The *Upper Confidence Bound* (UCB) algorithm is a standard approach to the MAB problem, that follows the exploration-exploitation idea, but additionally estimates the *confidence* in the exploiting selection. Instead of using the ϵ parameter, the approach suggests to define A_t as

$$A_t = \arg\max_{a \in A_t} \left(Q_t(a) + \delta \sqrt{\frac{2 \ln t}{k_t(a)}} \right)$$

where $k_t(a) = \sum_{i=1}^{t} [a = a_i]$, and δ is another parameter for exploration-exploitation trade-off. The value that is maximized is called the upper confidence bound, and is larger for actions that either have large $Q_t(a)$ value, or have a low number of occurrences $k_t(a)$.

Two examples of RAN where UCB is applied are the Global Dynamic Pricing problem, covered in Section 4.6, and the *Online Bipartite Matching* (OBM) problem. OBM is a generalization of the Bipartite Matching problem to a discrete-time domain when nodes arrive as time passes, with the arrival time unknown in advance. Once a node arrives, an OBM algorithm should provide immediate and irrevocable decision on the matching of the node. An example of OBM is a problem of dispatching drivers in a Ride-Hailing platforms [192, 265]. As soon as a customer appears, an algorithm should match him with one or several drivers. OBM can be solved by LP [192], online convex optimization methods [111], the UCB algorithm [39], and the Reinforcement Learning approach [265], which we cover in the next section.

5.2 Reinforcement Learning

Reinforcement Learning (RL) is one of the most active research areas in DS, having over 6000 publications just in January 2020, according to Google Scholar. It is a distinct Machine Learning paradigm, alongside with Supervised and Unsupervised learning, which follows the idea of goal-oriented learning from interaction [244]. Unlike the MAP problem, a general formulation of an RL problem allows an agent to *influence* the environment. RL techniques are useful when the problem is data-intensive, and the environment and transition dynamics are too complex to be described as a detailed stochastic model. This is a major difference with OR and Stochastic Control methods, such as Dynamic Programming. Modern RL is tightly coupled with Artificial Intelligence, since it is a common practice is to parametrize a policy with *Neural Network* (NN), and train NN using a simulator of the environment. For a historical overview and an all-inclusive survey of existing RL methods, refer to the book of Sutton and Barton [244].

RL usually implies estimating *a value function*, which evaluates the future profitability of a state. Therefore, a trained model implies *planning* a course of action, and *predicting* the future outcomes before they are experienced. This is a core difference between RL and *evolutionary* methods, such as genetic

algorithms and Simulated Annealing, used in OR as heuristics. Evolutionary methods can be applied to RL problems, and are effective if a search space for an optimal policy is small, or when an agent cannot accurately sense its state. Such methods do not use value functions and usually are out of the scope of RL literature [244].

Two basic RL methods are *Q-learning* and *Actor-Critic*. Most of the existing algorithms are an advanced version of either. A simple Q-learning algorithm is an off-policy Temporal-Difference learning algorithm that trains an action-value (Q) function. Let r_t be a reward at time step t, α and γ – learning parameters. The Bellman equation for Q-learning is [244]

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

$Q(s_t, a_t)$ is the *action-value function* because it shows profitability (*return*) of the state s_t *and* the action a_t. Q-learning is the *Temporal-Difference algorithm* because it updates the function after each time step and it is *Model-Free*, hence does not need to know a complete set of states and actions. Temporal-Difference methods are a fusion of Dynamic Programming and Monte-Carlo methods. Finally, Q-learning is *off-policy*, because at the learning phase it assumes actions are selected greedily according to the current Q-function, while in fact actions might be selected by a non-greedy policy. Q-learning where a policy is parametrized by Neural Network is referred as Deep Q-Network (DQN).

Actor-Critic methods imply a probabilistic policy. Unlike Q-learning, the action space is continuous. The policy is parametrized, and parameters are learned using gradient descent. The name comes from a collaboration of *an Actor*, a function that updates parameters of policy, and *a Critic*, a function that suggests an actor a direction of an update. Both Actor and Critic are usually parametrized by Neural Network. In Chapter 9 we consider an extension of the Actor-Critic method, namely cA2C, that uses a concept *a context*, a spatial-based collaboration method that restricts an effect of redundant swapping of locations among agents.

In this work, we are particularly interested in Multi-Agent RL (MARL), and its application to the Taxi Management problem. cA2C is one example of a scalable MARL solution. A comprehensive survey is proposed by Nguyen et al [196]. MARL can have cooperative or non-cooperative setting [167]. The collaborative setting is when agents explicitly learn to cooperate or compete. Such methods are unscalable, supporting up to a dozen of agents [293], because of *the credit assignment problem*, a common issue in RL when rewards from an environment are too sparse or too far in time for a model to associate with a cause. The non-cooperative setting is when the interaction of the agents are a part of a model. Most of the approaches are an extension of DQN, and only a few are extensions of Actor-Critic methods [196]. cA2C is a non-collaborative large-scale method designed specifically for taxi fleet management.

One of the latest innovations in MARL is the Proximal Policy Optimization (PPO) algorithm with the Intrinsic Curiosity Module [204]. PPO is a state-of-the-art single-agent policy optimization algorithm that extends Actor-Critic by additional regularization of policy parameters. The curiosity is a measure of surprise the encountered state brings to the agent. The architecture consists of two models, one predicts the next state of an environment, another that predicts an action that should correspond to a transition between the current state and the next state. The *intrinsic* reward is calculated as a distance between the actual next state and predicted next state. As a result, the states that are most unpredictable are valued the most. The algorithm shows extraordinary results in environments extremely sparse rewards, such as walking in mazes [259], and Multi-Agent games with cooperation [204]. For the recent overview on cooperative game-like MARL, refer to Zhao et al. [293], and a blog of OpenAI[1], a major AI research laboratory, authors of PPO, Gym[2] (a toolkit and a collection of standard baselines for RL), OpenAI Five[3] (an expert-level AI for one of the most complex multi-player game DOTA2 [217]), and GPT-2[4] (a Natural Language Processing model, trained on 8 million web pages, and capable of writing human-level quality fake news).

In this thesis, we study a new objective that maximizes the minimum revenue of drivers and adapts several RL solutions to the fair objective in a multi-agent setting. Training for the fair objective, in general, is more difficult than the total reward objective, since rewards are defined by the poorest driver only, so they are even sparser. We address the problem heuristically. Chapter 9 discusses the results. We use PPO as one of the baselines, and leave the embedding of the curiosity in our fair taxi management algorithms for future work.

[1] https://openai.com/
[2] https://gym.openai.com/
[3] https://openai.com/projects/five/
[4] https://openai.com/blog/better-language-models/

Chapter 6

Diffusion in Networks

Resource allocation in social networks or computer networks requires a special approach due to the crucial differences in network topologies, the behaviour of participating agents, their interaction, and, consequently, probabilistic models. Unlike FL, where the objective is defined by a cost of flow or transportation along a specific path, such phenomena as a disease spread require to take into consideration all possible paths, since an infected node might influence any number of adjacent nodes, so the flow preservation does not hold. In our study we model diffusion as independent cascades, and consider problems of Influence Maximization, Node Immunization, and Robustness of Cascade Diffusion under Node Attacks, with an emphasis on information spread control in social networks.

6.1 Influence Maximization

The diffusion model we focus on in this thesis is the *Independent Cascade* (IC) model, which is a descendant of compartmental models, i.e. mathematical models of infectious diseases. It assumes that each edge is labeled by a probability of *influence* through the edge, and applied mostly in a context of information diffusion in social networks. We provide the precise definition of IC in Chapter 10 of this thesis. The model became especially famous after the seminal work of Kempe et al. [138], where they propose to model information diffusion in social networks as IC, and formulate the *Influence Maximization* problem (IM), which we consider a special case of RAN. The objective of the IM model is to select *seeds* – nodes that initiate the propagation. The problem is NP-hard. The key observation of the authors is that the objective of IM is submodular, so the problem can be solved greedily, in-scale and with an approximation guarantee.

The IM problem proved to be effective in viral marketing and became popular in the DS community due to the countless possible IM extensions where the submodularity property preserves. We list some of them. IM may include

user preferences [131], topic-awareness [21], dynamic seeding [251], negative (competitive) opinion spread [44, 277]. *User preferences* are represented as additional edges in a network with weight function induced by collaborative filtering. *Topic-aware* diffusion model, similarly to the model with user preferences, is based on calculating a vector product of a propagated item and an edge label. *Dynamic seeding* [251] model implies that new seeds are added iteratively. Consider Li et al. [165] for a complete survey.

6.2 Diffusion and the Facility Location problem

In order to illustrate a connection of the IM problem with the Facility Location problem, we consider a toy example of RAN in a linear graph, and describe a polynomial time solution using Dynamic Programming. For the FL problem, let d_{ij} denote a matrix of geodesic distances. For the IM problem, let $0 < p_{i,j} \leq 1$ be a probability that the edge (a_i, a_j) survives, and a matrix d_{ij} contain probabilities of paths existence between each pair of nodes:

$$d_{ij} = \prod_{k=i+1..j} p_{k-1,k}$$

Eqs. 6.1 and 6.2 show the Bellman equations for the FL and IM problems. Here, $f_{\text{FL}}(m, k)$ says that the cost of the assignment of the k-th facility (state variable) to node m (control variable) is equal to the minimum cost of the assignment of the $(k-1)$ facility to node j, plus the cost of adding a new facility to the m-th node (payoff function). The minimization occurs over control variable j. The cost of the new facility is equal to the sum of distances to the closest facility over all nodes in the range $(j \dots m)$. Next, $f_{\text{IM}}(m, k)$ has similar meaning, with the payoff function shown in Eq. 6.3.

$$f_{\text{FL}}(m, k) = \min_{1 \leq j < m} \left\{ f_{\text{FL}}(j, k-1) + \sum_{i=j+1}^{m-1} \min\{d_{ji}, d_{im}\} \right\} \tag{6.1}$$

$$f_{\text{IM}}(m, k) = \max_{1 \leq j < m} \{ f_{\text{IM}}(j, k-1) + S(j, m) \} \tag{6.2}$$

$$S(j, m) = \sum_{x=j}^{m} (d_{jx} + d_{xm} - d_{jx} d_{xm}) \tag{6.3}$$

The base case in for both problems appears when $m = k$. Then, each of the m nodes contains a resource, and the value of the functions is just the sum of distances $\sum_{1 \leq i < m} d_{i(i+1)}$. The optimal the objective for locating k facilities in n nodes can be found by the recursive calls

$$f_{\text{FL}}(k) = \min_{m \in [k \dots n]} f_{\text{FL}}(m, k)$$

$$f_{\text{IM}}(k) = \max_{m \in [k...n]} f_{\text{IM}}(m, k)$$

Under the assumption of small influence probabilities [279], we can also linearize the IM objective and provide an LP formulation for the toy RAN under the diffusion-type interaction, similarly to the LP formulation for the facility location problem (Eq. 4.1). The probability of a node i being active in a linear graph is equal to

$$S(i) = 1 - (1 - d_{li})(1 - d_{ir}), \tag{6.4}$$

where l and r are indexes of the left and right closest seeds, respectively. The expected number of active nodes is the sum of $S(i)$ over all nodes. We can express d_{li} as $\sum_{j=1}^{n} d_{ji} \cdot y_{ji}$ with the constraint that $\sum_{j=i+1}^{n} y_{ji} = 0$, meaning that we assign only to seeds with a smaller index. One new variable $z_{ij} \in \{0, 1\}$ is used for assignment to a seed with a larger index in a similar manner. We can linearize the objective by taking into consideration that probabilities of information propagation in real-world graphs are small [279]. The resulting LP objective is presented in Eq. 6.5 with the corresponding constraints occurring in Eqs. 6.6,6.7,6.8.

$$\max_{y_{ij}, z_{ij}} \sum_i \sum_j d_{ij} y_{ij} + \sum_i \sum_j d_{ij} z_{ij} \tag{6.5}$$

$$y_{ij} \leq x_j, \quad z_{ij} \leq x_j, \quad \sum_j x_j = k \tag{6.6}$$

$$\forall k \quad \sum_{i=0}^{k-1} y_{ik} \leq 1, \quad \sum_{i=k+1}^{n} z_{ik} \leq 1 \tag{6.7}$$

$$\forall k \quad \sum_{i=k+1}^{n} y_{ik} = 0, \quad \sum_{i=0}^{k-1} z_{ik} = 0 \tag{6.8}$$

LP can also provide solutions to the IM problem for arbitrary graph topologies [238, 238]. However, such solutions have limited scalability. For example, Sing and Dinh [238] presents a mixed LP and randomized algorithm that can process a network up to 1.5K nodes and 2.7K edges within a time limit of 1h.

6.3 Scalable Spread Control

Data-Intensive solutions to IM can be categorized into Monte-Carlo-based, Proxy-based and Sketch-based approaches [165]. Proxy-based approaches propose practically effective proxy models to approximate the IM objective. Sketch-based approaches use simulations similarly to the Monte-Carlo methods, but collect a targeted experience with an emphasis on theoretical efficiency. Sketch-based approaches are known to be both efficient (unlike Monte-Carlo methods) and provide the approximation guarantees (unlike Proxy-based solutions). Most scalable Sketch-based approaches are based on the

idea of Reverse Reachable (RR) Sets, initially presented by Borgs et al. [36], and improved by Tang et al. [246]. Given a random node v in a graph and a world outcome, a reverse reachable set is such a set of nodes, that activation of any node in that set would lead to activation of v. Sampling v and world outcomes lead to a collection of RR sets. Then, a node that is a member of the largest number of RR sets is the most influential. The idea of RR sets is somehow close to the idea of Inverted Indexing [145] and Bichromatic Reverse Nearest Neighbor (BRNN) technique [52, 282], used in Optimal Location Queries. We talk about the BRNN technique in Chapter 8, and use RR-based solutions in Chapters 10 and 11 of the thesis.

The problem dual to IM is the *Node Immunization* (NI) problem. Originating from the field of immunology [69, 234], the problem is to allocate a limited amount of vaccines in a network so to suppress a virus epidemics. We consider this problem in Chapters 10 and 11 of the thesis. IM and NI together form a larger class of problems of Diffusion Spread Control [227], and Network Robustness, which we study in Chapter 11.

Chapter 7

Robustness and Fairness

Our special interest in RAN is robustness and fairness of solutions. A solution is *robust* if it preserves high effectiveness under adversarial perturbation of an input data. The robust version of RAN can be seen as a two-player game, where the role of a second player is an adversary that designs the perturbation [114]. The problem is formulated with a minimax objective

$$\max_{\Omega} \min_{\Phi} f(\Omega, \Phi)$$

where Ω is a set of possible choices for the player who seeks a robust outcome, Φ is a set of choices of the adversary, and $f(\Omega, \Phi)$ is a set function that defines an outcome of the game. In 1928 John von Neumann published the first minimax theorem [260]. It states that for any compact convex sets Ω and Φ, and a concave-convex function f,

$$\max_{\Omega} \min_{\Phi} f(\Omega, \Phi) = \min_{\Phi} \max_{\Omega} f(\Omega, \Phi)$$

The theorem is considered as the starting point of Game Theory. A solution where none of the players can benefit from changing their own strategy is called the *Nash Equilibrium*. A two-player zero-sum game can be formulated as an LP program, and the Nash equilibrium corresponds to the optimal solution of the primal and dual problems [66]. The notion of equilibrium was expanded to more complex games and decision-making problems with uncertainty, where the equilibrium might not exist. A question of finding the equilibrium in different types of games is an active topic in DS, and Multi-Agent Reinforcement Learning in particular [88, 213].

7.1 Fairness as a Form of Robustness

The Nash Equilibrium has been linked to the notion of *fairness* [88]. Seen as an optimization problem, fairness also appears with a maximin objective maximizing the worst-case profitability among agents [159]. Hence fairness

in this sense is another name for robustness. Other concepts of fairness exist [215], such as minimization of the difference in profitability among any two agents [70, 212, 243]. A more detailed review is provided in Section 9.2.

7.2 Fairness and Robustness in Resource Allocation

In this thesis, we study the problems of robustness and fairness of RAN with flow-type and diffusion-type interactions. For the flow-type interaction, we note few works on Robust FL [107, 236], fair FL with pricing [274], and concentrate on fair RAN under MDP framework in Chapter 9. Specifically, we consider the optimization of driver-oriented fairness in the taxi cruising problem. Nikulin et al [197] provides a bibliography with robust flow-type combinatorial optimization problems.

In 2017, Jegelka and Staib [239] suggested to apply submodular minimization in the problem of Robust Budget Allocation. Interestingly, the work puts two types of consumer-resource interaction under one umbrella of submodular optimization. The problems mix Stochastic Bipartite Matching, FL and IM. The goal is to allocate a budget over a stochastic bipartite graph, where a budget for a node indicates how many attempts the node has to influence the target. The solution is based on submodular minimization and gradient descent.

For the diffusion-type interaction, we study a two-player game, where one player solves the IM problem, another - the NI problem. The maximin objective in the context of IM was first proposed by He and Kempe in 2016 [114]. In Chapter 11, we expand their study with additional experiments and propose to use the robustness objective as a measure of robustness of a stochastic network. Then, we propose 3 novel robustness measures, depending on a sequence of maximum, minimum and expectation operators in the objective, and study how robustness of different network types depends on the sequence. The expectation is over diffusion outcomes. Our study is seeder-oriented, and a particular sequence depends on a seeder *awareness* about an environment and the opponent. Unlike the work of Zhang et al., we also vary the position of the maximization operator.

7.3 Connections to Other Areas

Our study relates to the study on Robust Neural Networks [297]. A neural network adversarial attack (NNAA) occurs before or after training; this distinction is similar to our distinction of attacks before or after diffusion, namely causative or exploratory attacks. Seed selection in the Robust IM context is similar to the selection of the node to classify in the NNAA context. The

training process of a neural network is an analogy of the expectation operator, a function that brings uncertainty due to its nature or computational intractability. Our study suggests a new interesting research question, where seeding (selection a node for classification) is done after the attack. The solutions of Zuenger et al. [297] are based on the "surrogate model", a linearized model, similarly to the Proxy-based techniques in IM literature.

Another relevant study was presented by Zhang et al [286]. Authors consider decision making under the Multi-Agent Markov Decision Process framework, and provide several definitions of fairness depending on the position of the expectation operator in a maximin objective.

Part II

Publications

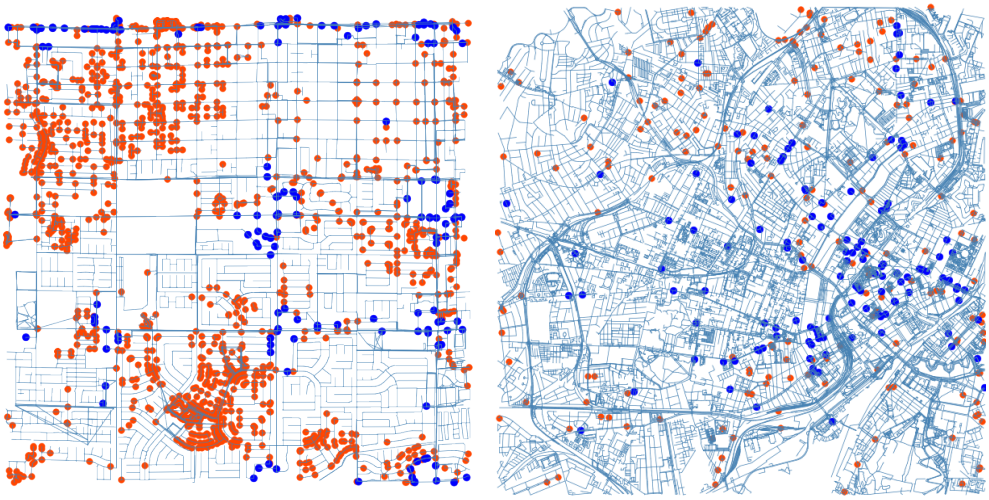
Chapter 8

Multicapacity Facility Selection in Networks

Consider the task of selecting a set of facilities, e.g., hotspots, shops, or utility stations, each with a *capacity* to serve a certain number of customers. Given a set of customer locations, we have to minimize a cumulative *distance* between each customer and the facility earmarked to serve this customer within its capacity. This problem is known as the Capacitated k-Median (CKM) problem. In a data-intensive variant, distances are calculated over a network, while a data set associates each candidate facility location with a different capacity. In other words, going beyond positioning facilities in a metric space, the problem is to select a small subset out of a large data set of candidate network-based facilities with capacity constraints. We call this variant the Multicapacity Facility Selection (MCFS) problem. Linear Programming solutions are unable to contend with the network sizes and supplies of candidate facilities encountered in real-world applications; yet the problem may need to be solved scalably and repeatedly, as in applications requiring the dynamic reallocation of customers to facilities.

We present the first, to our knowledge, solution to the MCFS problem that achieves both scalability and high quality, the Wide Matching Algorithm (WMA). WMA iteratively assigns customers to *candidate* facilities and leverages a data-driven heuristic for the Set Cover problem inherent to the MCFS problem. An extensive experimental study with real-world and synthetic networks demonstrates that WMA scales gracefully to million-node networks and large facility and customer data sets; further, WMA provides a solution quality superior to scalable baselines (also proposed in the chapter) and competitive vis-á-vis the optimal solution, returned by an off-the-shelf solver that runs only on small facility databases.

The content of this chapter was published in Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE 2019), in co-authorship with Panagiotis Karras and Christian S. Jensen [173].



(a) Las Vegas (b) Copenhagen

Figure 81: Customers (red), and cafés (blue).

8.1 Introduction

A type of problem arising in smart city applications calls for selecting an attractive subset out of a set of candidate *facility locations* (e.g., telecommunications hotspots, meeting points, bike stations, recycling stations, electric vehicle charging stations, or waste disposal sites) to provide a service in an urban network. The fitness of the selected set is measured by means of total convenience or utility with respect to a set of geographically located *customers*. This problem may need to be solved repeatedly; for example, one may need to periodically decide on a set of service locations, depending on which customers declare interest for a certain offering.

The input data typically includes a weighted graph, representing a road network and associated candidate facility locations and customer locations. Figure 81 provides two examples, where we need to select a subset out of a set of eligible facilities (in *blue*) so as to serve a predefined set of customers (in *red*) in Las Vegas or Copenhagen.

In the *Multicapacity Facility Selection* (MCFS) problem, each candidate facility has a *capacity*, and we need to choose k facilities and assign each customer to one of them while observing the capacity constraints; as the number of served customers is bounded by capacity constraints, an objective of maximizing that number does not arise; the objective is to optimize a notion of *customer convenience*, defined in terms of the distances between customer and the facilities they are assigned to. The MCFS problem amounts to the *hard* and *nonuniform* case of the *capacitated k-median* (CKM) problem [178] over a *network*. Here, *hard* indicates that only one facility can be placed at

a certain location; the *soft* version allows for multiple facilities at the same location. Next, *nonuniform* indicates that facility capacities differ; in the *uniform* version, all capacities are equal. Last, the *network* setting need not yield a metric distance notion.

Unfortunately, the problem is already NP-hard in the *soft* and *uniform* case over a *metric space* [146]. Small instances can be solved exactly by Linear Programming (LP) and Mixed Integer Programming (MIP) solvers [109]. However, such solvers do not scale beyond networks with a few thousand nodes. Past research has proposed LP *relaxation* [84] methods that provide approximation guarantees while violating constraints on facility *capacity* or *cardinality*. The most recent works in the area introduce an LP formulation, called *rectangle LP*, tailored to the uniform [163] and soft nonuniform [161] capacitated k-median problems, and develop rounding algorithms that achieve constant approximation guarantees while violating the cardinality constraint k. Such solutions remain impracticable in real-world applications due to their high-polynomial time complexities, while an approximation algorithm for the nonuniform hard-capacitated case has yet to be developed [162].

Local search techniques exist for the CKM problem and related *facility location* problems [57, 146], known as *group nearest group queries* in the database community [74]; however, such solutions solve only the *uncapacitated* and *uniform soft-capacitated* problem cases; they accommodate neither nonuniform nor hard capacity constraints. Thus, to our knowledge, no existing solution achieves both high quality and scalability to large networks and customer sets in the MCFS problem.

We present an effective and scalable MCFS solution, the *Wide Matching Algorithm* (WMA). WMA progressively assigns customers to strategically chosen candidate facilities, translating a *bipartite assignment* under capacity constraints to a network setting, and decides on its termination by means of a Set Cover heuristic. We contribute the following:

- We attempt the first, to our knowledge, solution for the MCFS problem that achieves high quality and is applicable to large real-world networks.
- We develop an algorithm, WMA, that combines a data-driven heuristic for set cover with a principled spatial assignment subroutine.
- We introduce a reasonable baseline MCFS heuristic that clusters customers in groups satisfying capacity constrains, following a Hilbert space-filling curve.
- We conduct an experimental study with synthetic and real data, demonstrating that WMA scales to million-node and million-edge networks with large customer and facility sets and achieves near-optimal solution quality, as seen in cases where the exact solution can be computed.

8.2 Problem Statement

Consider a network represented as a weighted (directed or undirected) graph $G = (V, E, W)$, where V is a set of nodes that model urban locations such as intersections and road ends, E is a set of edges that model road segments, and W is a mapping from edges to positive integer weights that model road segment lengths. Further, we are given a set of m *customers* $S = \{s_i\}_{i=1}^{m} \subseteq V$, and a set of ℓ *candidate facility locations* $F_p = \{f_j\}_{j=1}^{l} \subseteq V$; each $f_j \in F_p$ comes with a capacity constraint c_j. Given a cardinality value k, the problem is to select k candidate facilities $F \subseteq F_p, |F| \leq k$ and *assign* each customer to *exactly one* facility in F, so that each selected facility $f_j \in F$ is at most c_j assigned customers and the sum of network distances between customers and their allocated facilities is minimized.

We use two binary variables x_j and y_{ij}; x_j indicates whether the candidate facility at node v_j is selected, $j \in \{1..\ell\}$, while y_{ij} indicates whether the customer at node v_i is assigned to the facility at node v_j. Also, let d_{ij} be the shortest-path distance between v_i and v_j. Note that d_{ij} values need not define a metric matrix and need not be given as input; instead, they may be computed on the fly over the input network, a feature distinguishing our problem setting. Then, our minimization objective over x_j and y_{ij} is:

$$\min_{x_j, y_{ij}} \sum_i \sum_j d_{ij} y_{ij} \tag{8.1}$$

subject to:

$$y_{ij} \leq x_j, \quad x_j, y_{ij} \in \{0, 1\} \tag{8.2}$$

$$\sum_j y_{ij} = 1, \quad \sum_i y_{ij} \leq c_j, \quad \sum_j x_j \leq k \tag{8.3}$$

Constraint (8.2) implies that a customer can be assigned to a node v_j where a selected facility is located. The other constraints stipulate that each customer is assigned to *exactly one* facility, a facility v_j is matched with *at most* c_j customers, and k facilities are selected. Table 81 outlines our notations.

8.3 Related Work

An array of facility location problem variants have attracted attention for a long time. Farahani and Hekmatfar [84] provide a comprehensive overview of state-of-the-art algorithms for several of those variants. These algorithms are mostly based on *linear programming* (LP), using *LP-rounding* and *Lagrangian relaxation* as approximation tools. The problem we study is a network-based version of the *nonuniform hard-capacitated k-median* problem [67, 163, 178].

Notation	Description
G	A weighted graph (network)
E, V	Sets of edges and nodes in G
$v.dist$	Distance from considered customer to node v
$v.p$	Potential of node v
$dist(v_1, v_2)$	The shortest path distance between nodes v_1 and v_2 in G
$S \subseteq V$	Locations of customers
n	Number of nodes in G, $n = \|V\|$
m	Number of customers
k	Number of selected facilities
ℓ	Number of candidate facilities
c_j	Capacity of facility j
$F_p \subseteq V$	Set of candidate facility locations
$F \subseteq F_p$	Selected facilities, $\|F\| = k$
G_b	Bipartite directed graph between C and F_p
E'	Set of edges in G_b
d_{ij}	Distance between i-th customer and j-th candidate facility
$x_j \in \{0, 1\}$	Indicator of whether f_j is in F
$y_{ij} \in \{0, 1\}$	Indicator of whether s_i is allocated to f_j
d_i	Demand of a customer s_i in bipartite graph G_b
σ	Assignment of customers to facilities in G_b
$\sigma_j(G_b)$	Set of customers assigned to facility $f_j \in G_b$

Table 81: Notations.

Scalable Facility Location

Some recent works consider a special facility location problem variant, called Optimal Location Query (OLQ) [52, 57, 280, 282], which calls to place a *single* new facility that attracts the highest amount of customers (the *MaxSum* objective), or minimizes the maximum distance between a customer and its nearest facility (the *MinMax* objective).

OLQ solutions are based on a *Bichromatic Reverse Nearest Neighbor* (BRNN) technique, where each customer is associated with a *Nearest Location Region* (NLR), such that each point therein is closer than the nearest *existing* facility. To optimize for MaxSum, we place a new facility in the region with the highest amount of overlapping NLRs [52, 282]. To optimize for MinMax, we sort customers by distance to nearest facility, obtain a set of top-k customers whose NLRs' intersection is nonempty, and find an optimal region therein [57].

As the OLQ bears some resemblance to to the MCFS problem, we could apply it *iteratively*, as a heuristic, to obtain a solution to MCFS. Figure 82 illustrates the result of facility selection by such an approach, employing the intuitively reasonable MaxSum objective. We start with no facility placed and select node 1 for the first facility, as it is the one that minimizes the aggregate

distance to customers a, b, c. Dashed curves in the figure indicate the resulting NLRs for each customer. Node 2 has the highest number of intersecting NLRs (i.e., attracted customers), so we select it. Yet the optimal MCFS solution is to select nodes 4 and 5. Thus, unfortunately, placing facilities by an iterative BRNN-based approach does not fare well with our optimization objective.

We implemented a BRNN-based approach that sequentially selects k nodes as facilities, recalculating a set of NLRs at each step and breaking ties arbitrarily. We include this approach in our experimental comparison; as we will see, its results are significantly worse than those of other approaches.

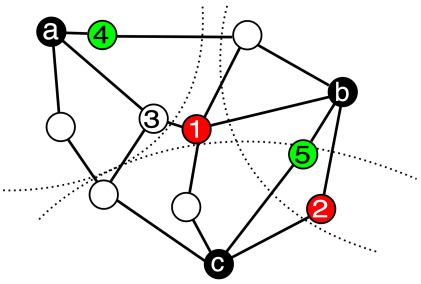


Figure 82: BRNN application

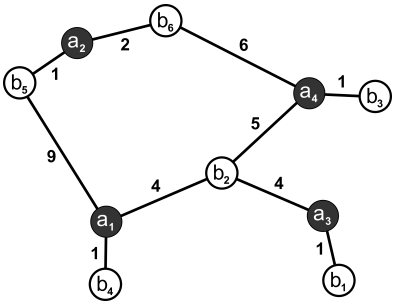


Figure 83: Example network G

Bipartite Matching

The MCFS problem implies a bipartite matching of customers with facilities. To address this need, we adapt the *Simplified Incremental Algorithm* (SIA) [256, 257] from the case of Euclidean distances to that of network distances. SIA adapts the *Successive Shortest Path Algorithm* (SSPA) [75] to a bipartite graph, enhancing it with an *edge pruning* capability, which allows finding a provably optimal matching after accessing only a few edges adjacent to each node, using an edge weight threshold derived from *node potentials*. In Section 8.5, we enhance this pruning threshold.

8.4 The Wide Matching Algorithm

In a nutshell, the Wide Matching Algorithm progressively enriches candidate facilities with potential serviced customers until it finds a set of k facilities that can service the full customer set within their capacities.

Algorithm Overview

Throughout the operation of the algorithm, each customer s_i maintains an increasing *demand* value d_i, reflecting the number of candidate facilities in F_p it has to be assigned to. In each iteration, we increase the *demand* of a chosen

subset of customers and assign each customer with increased demand to *exactly one* new facility; while doing so, we may *rewire* previous choices, i.e., reallocate previous customer-to-facility allocations, if beneficial, while observing capacity constraints. Thereby, customers explore candidate assigned facilities, though eventually they are allocated to exactly *one* of those. We then select a subset $F \subseteq F_p$, $|F| = k$, such that the elements of F collectively *cover* (i.e., are allocated to within their capacities) as many customers as possible, by means of a SET COVER heuristic; this heuristic iteratively picks a facility that brings the biggest *marginal gain* to the number of covered customers. We resolve ties by selecting the facility f chosen *least recently* in previous iterations. This diversification strategy avoids getting trapped in non-optimal local minima. An *exploration vector* specifies the increase of d_i values per iteration: Δd_i is set to 1 *if and only if* s_i has been left *uncovered* by the set F selected in the previous iteration and $d_i < \ell$; this choice lets all customers grow their demand values evenly. The main phase WMA terminates when it detects a subset $F \subseteq F_p$ that covers *all* customers in S, or all demands reach ℓ; the latter case invokes special measures, which we discuss in Section 8.4.

Example

We illustrate the operation of WMA with an example. Figure 83 shows a network of 9 nodes, a_i for customers and b_j for candidate facilities. For visualization's sake, we do not place facilities on the same nodes as customers.

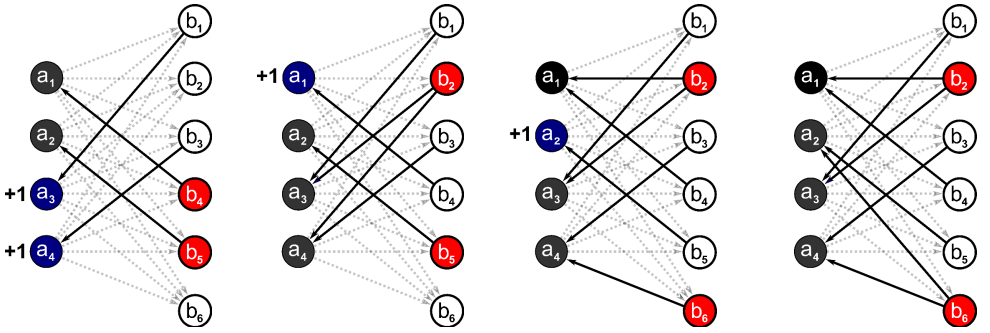
Assume we have to place $k = 2$ facilities, with uniform capacity $c = 2$. Figure 84 shows the bipartite graph G_b from customers to candidate facilities across iterations. Each edge is weighted by the distance between its adjacent nodes. Table 82 depicts part of the adjacency list of G_b with each node's three nearest adjacent nodes in ascending order.

a_1	$b_4(1)$	$b_2(4)$	$b_5(9)$
a_2	$b_5(1)$	$b_6(2)$	$b_3(9)$
a_3	$b_1(1)$	$b_2(4)$	$b_4(9)$
a_4	$b_3(1)$	$b_2(5)$	$b_6(6)$

Table 82: Sample adjacency list for G_b; weights in brackets.

First, each customer is matched to its nearest facility in Figure 84a. Now each of the four facilities covers one customer. We resolve ties arbitrarily, selecting two facilities out of four, b_4 and b_5, and set the exploration vector to $\Delta d = \{0, 0, 1, 1\}$.

In effect, a_3 and a_4 need to explore the network further. They do so and both acquire a new match, facility b_2, obeying the capacity constraint $c = 2$. Thus, by the end of the second iteration, a_1 and a_2 have been matched to one facility each, while each of a_3 and a_4 has been matched to two facilities. Now



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 84: Bipartite graph G_b through WMA iterations.

b_2 is the *most popular* facility, in the sense that it is matched to more customers than any other facility, namely customers a_3 and a_4. After discounting these covered customers, the second most popular facility in terms of marginal gain is either b_4 or b_5, bringing a gain of one customer each, i.e., a_1 and a_2, respectively. We arbitrarily select one of the two, b_5. At this point a_1 is the only *uncovered* customer. Hence, we set $\Delta d_1 = 1$ and $\Delta d_i = 0$ for the other three customers, as Figure 84b illustrates.

The third iteration (Figure 84c) demonstrates the benefit of using an assignment algorithm. Now a_1 has a demand to be matched with two facilities, yet its next nearest facility, b_2, has reached its capacity; a greedy approach would then match a_1 to b_5, the next nearest available facility. Rather than doing so, our matching algorithm *rewires* previous choices, i.e., reconsiders previous allocations and substitutes them with new ones, if beneficial: in particular, it reassigns a_4 to b_6 so that it can assign a_1 to b_2. The newly used b_6 along with b_2 collectively cover a_1, a_3, and a_4. Now a_2 is uncovered, and hence $\Delta d_2 = 1$. Eventually, the fourth iteration matches a_2 to b_6. Now two facilities, namely b_2 and b_6, cover all customers, as Figure 84d shows, with objective value 16.

Algorithm Outline

WMA operates on a *complete directed* bipartite graph between customers and candidate facilities, and progressively satisfies demand and capacity constraints by bipartite matching. This operation can be time consuming on a complete graph, while previous work has not considered bipartite matchings among nodes anchored in a network. Still, we effectively transfer a pruning technique for bipartite matching with Euclidean distances [256, 257] to a network setting.

The core idea is this: if we can ascertain that there is no possible beneficial reassignment that would match node a_i to b_j, we can eschew b_j from consid-

eration. To ascertain that, we do not need to know the *exact* weight of edge (a_i, b_j); it suffices to know that b_j is farther than another possible match, b_k. We can expand knowledge of such weights incrementally on demand, running an instance of Dijkstra's algorithm on G per customer in each iteration.

Algorithm 1 outlines WMA. In each iteration, we first try matching with current customer demands (Lines 4–5); then we check whether we can select a set of facilities F that cover all customers (Line 6); if we cannot, we raise demands appropriately (Lines 7–8). Lines 10–11 cover the special case where there exists a set F such that $|F| < k$ and F already covers all customers. In that case, we locate the remaining $k - |F|$ facilities in the vicinity of customers with the most unsuccessful assignments; this measure retains coverage and improves the cost objective. Algorithm 4 in Section 8.4 illustrates this process. In case the k selected facilities fail to cover some customers even after their demands reach ℓ, Lines 12–13 revise F ensuring it suffices to cover all customers, i.e., all disconnected network components. Algorithm 5 in Section 8.4 provides the details. Eventually, Lines 14–15 call the same process recursively, setting the demand of each customer to 1, so as to build a single *optimal-cost* assignment, $\sigma(G_b)$, of customers to the k selected facilities in F; the edges in $\sigma(G_b)$ outgoing from a selected facility f_j define the set of customers $\sigma_j(G_b)$ matched to f_j.

WMA maintains two graphs throughout its operation: first, the input network G that contains locations of customers and candidate facilities; second, the bipartite graph G_b, used for extracting assignments among those entities. Edge weights in G_b reflect shortest-path distances between customers and facilities in G. We assume that a single facility can be located on any network node; the algorithm can be straightforwardly extended to any restrictions on such placements by tuning the candidate facility nodes in G_b.

Matching Function

Let us discuss the matching function that iteratively assigns new customers to facilities in G_b and reassigns previously matched pairs. The complete bipartite graph G_b has $\ell \cdot m$ edges, where each edge requires an execution of Dijkstra's algorithm for its weight calculation. For large problem instances, that would cause excessive computation. Therefore, we add edges to G_b only on demand.

We initialize G_b with two sets of nodes: customers and facilities, without edges. We add edges progressively, running a variant of the *Successive Shortest Path Algorithm* (SSPA) [75] with node *potentials*; such potentials encapsulate the goodness of the current arrangement for a node in question, so that we can calculate the benefit of updates involving that node. The process terminates when we can guarantee that the running matching is optimal in the complete G_b. The SSPA solves the *Minimum-Cost Flow* problem (to which bipartite matching is reduced) using iterative Dijkstra executions from a *source* to a *sink*, and *flow augmentation*. In our problem, the source is a customer s, the

Algorithm 1 Wide Matching Algorithm

```
1: function LocateFacilities(G, S, F_p, k)
2:     G_b ← Bipartite empty graph based on G
3:     d_i = 1   ∀i
4:     repeat
5:         for all s_i ∈ S : d_i > |{f_j | s_i ∈ σ_j(G_b)}| do
6:             G_b ← FindPair(G_b, s_i)
7:         {F, Δd, covered} ← CheckCover(G_b, k)
8:         d ← d + Δd
9:     until ∀i Δd_i = 0
10:    if |F| < k then
11:        SelectGreedy(F, G)
12:    if not covered then
13:        F ← CoverComponents(S, F, G)
14:    if |F_p| > k then
15:        return LocateFacilities(G, S, F, k)
16:    else
17:        return F, σ(G_p)
```

sink is the closest *non-fully occupied* facility in G_b; flow augmentation amounts to substituting an edge with one of opposite weight (given that a customer can be matched to each facility only once). SSPA guarantees optimality by adding new edges in an order sorted by weight: it maintains the running weight of the next *candidate edge* to be taken into consideration, and derives a threshold indicating whether that weight can affect the current solution. We discuss this threshold in Section 8.5. We achieve this order by one Dijkstra execution per customer, yielding distances to candidate facilities in non-decreasing order; such distance values give the weights of new edges in G_b.

Algorithm 2 presents the pseudocode for matching a customer in G_b and updating the running assignment by *rewiring* as necessary. The loop of Lines 4–12 adds edges to G_b until it can accept a new match for the given customer. In each iteration, we run a Dijkstra instance on G_b (Line 8), to find a shortest path in G_b from the given customer s to the nearest usable (i.e., not fully occupied) facility; this Dijkstra instance works with weights reduced by *potential* values $v.p$, and it returns the found *path* and the set of *visited* nodes. We add each visited node v to a *heap* with a *threshold* value that we justify in Theorem 1 (Section 8.5). This threshold depends on the distance from v to its next *nearest neighbor* in the network graph G ($nnDist$), the distance from *customer* to v in G_b ($v.dist$), and a *potential* value $v.p$ (Lines 9–11). When the condition in Line 12 is satisfied, we can proceed to update the running bipartite assignment.

Algorithm 2 Matching Function

```
1:  function FindPair(G_b, s)
2:      heap ← empty heap
3:      heap.add(⟨s, 0⟩)
4:      repeat
5:          x ← heap.topKey
6:          nn ← node in G_b for next NN of x in G
7:          add edge (x, nn) to G_b
8:          {path, visited} ← Dijkstra(s)
9:          for all v ∈ visited ∩ S do
10:             nnDist ← distance to next NN of v in G
11:             heap.add(⟨v, v.dist + nnDist − v.p⟩)
12:     until path.length < heap.topValue
13:     for all e ∈ path do
14:         e ← −e                                  ▷ Reverse edge
15:         w(e) ← −w(e)                            ▷ Reverse edge weight
16:     for all v ∈ visited do
17:         v.p ← v.p + path.length − v.dist
18:     return G_b
```

WMA runs two *independent* Dijsktra instances: one on the bipartite graph G_b (Line 8) for the sake of updating its running assignment and another on the network graph G (Line 10) for the sake of incrementally calculating edge weights on G_b. As both operate over graphs, they require no spatial index. The *path* found in Line 8 contains a *new match*, while observing capacity constraints. Then, the loop of Lines 13–15 performs *flow augmentation*: it increases the flow value by 1 along this *path* and performs necessary assignment and reassignment actions. Line 17 adjusts potential values.

One execution of the Matching Function assigns exactly one facility to one customer. The flow augmentation in SSPA is constrained only by the target's capacity and edge capacities; therefore, it is possible to augment flow by more than one in some cases. However, we do not need to do so, as we need *not* ever match the same customer with the *same* facility again. As we want many customers to be assigned to each facility, we set the *capacities* of edges in G_b to 1; thus, whenever time a customer is assigned to a candidate facility by FindPair, the flow is increased by 1.

Set Cover Routine

The need check whether we can select a subset F that covers all customers raises a Set Cover problem. As this problem is NP-hard, we employ a heuristic solution. After each iteration, we rank all candidate facilities by

their (dynamically updated) marginal gains and greedily select the top-k. If our selection covers all customers, WMA terminates. Algorithm 3 illustrates this approach. We place all candidate facilities in a heap, organized on the number of customers they cover, and extract facilities from the heap one by one, checking whether all customers served by the last extracted facility f remain uncovered. If so, we include the facility in our selection. Otherwise, we recalculate that facility's marginal gain and put it back in the heap. If we reach k facilities without achieving full coverage, we have not yet reached termination.

Algorithm 3 Checking top-k facilities

```
1:  function CheckCover(G_b, k)
2:      heap ← empty heap
3:      for all f_j ∈ F_p do
4:          f_j.m ← |σ_j(G_b)|
5:          heap.add(⟨f_j, f_j.m⟩)
6:      F ← ∅,   ∀i Δd_i ← 1
7:      for γ ∈ {1..k} do
8:          f_j ← heap.top
9:          m' ← |σ_j(G_b)|
10:         if f_j.m ≠ m' then
11:             f_j.m ← m'
12:             heap.add(⟨f_j, f_j.m⟩)
13:         else
14:             F ← F ∪ f_j
15:             for all {s_i | s_i ∈ σ_j(G_b) ∨ d_i = ℓ} do
16:                 Δd_i ← 0
17:             if ∀i Δd_i = 0 then
18:                 return F, Δd, true
19:     return F, Δd, false
```

Updating Demands

A crucial operation in WMA is the update of customer demands. A simple approach would increase the demand of all customers by 1 in each iteration. We have found that it is much more effective to increase the demand by 1 only for those customers that were not covered in the last iteration. This selective increase introduces those uncovered customers to more facilities, increasing the chances that they get covered sooner rather than later. Further, we keep track of how recently a facility has been used in a previous iteration to break ties between facilities that incur equal marginal gains.

Special Provisions

We have noted that Algorithm 1 (Section 8.4) makes provisions for two special cases: the case in which fewer than k facilities already cover all customers, and the one in which k facilities fail to cover some customers even after their demands reach ℓ. Here we describe these provisions.

Algorithm 4, called in Line 11 of Algorithm 1, provides the former special provision: it selects additional facilities until $|F| = k$. Each iteration of the main loop adds to F a new facility $f^* \in F_p \setminus F$ that is nearest to the customer s having the highest current distance to the nearest facility in F. Thereafter, Lines 14–15 in Algorithm 1 build an assignment using the enlarged F, yielding improved cost.

Algorithm 4 Greedy addition of facilities

```
1: function SelectGreedy(F, G_b)
2:     while |F| < k do
3:         s* ← arg max_s {min_{f∈F} dist(s, f) | s ∈ S}
4:         f* ← arg min_f {dist(s*, f) | f ∈ F_p \ F}
5:         F ← F ∪ f*
```

Algorithm 5 provides the latter special provision: it receives a set of selected facilities F as input and replaces facilities therein to ensure that each connected component of G is allocated sufficient capacity to cover all its customers. Line 3 calculates the difference $g.p$ between the collective capacity of selected facilities that are within connected component g, which we denoted as the set F_g, and the number of customers in g, $|S_g|$. A positive value of $g.p$ indicates that the facilities allocated to g by F suffice to cover the customers therein, with some possible reallocation. A negative $g.p$ means that component g should be offered more facilities or facilities with higher capacities. The loop in Lines 4–9 runs as long as a component with negative $g.p$ exists, substituting the lowest-capacity selected facility f in the highest-$g.p$ component g_M with the highest-capacity unselected facility in the lowest-$g.p$ component g_m. Theorem 3 proves that, if a solution exists, this loop terminates.

8.5 Matching optimality

Here, we prove that the FindPair routine of Section 8.4 yields an optimal assignment, even while using a simpler pruning criterion than the one in [257].

The sets of customers $S = \{s_i\}$ and facilities $F_p = \{f_j\}$ form the two sets of nodes in bipartite graph G_b. E'_f is the complete set of all possible edges of G_b, while E' is the set of edges that we are choosing to add to G_b. Also, $dist(s_i, f_j)$ is the weight of edge $(s_i, f_j) \in E'_f$; by the definition of G_b, $dist(s_i, f_j)$ is the shortest-path distance between customer s_i and facility f_j in graph G; $v.dist$ denotes the length of the shortest path sp from *customer*

Algorithm 5 Selecting facilities that cover all customers

```
1: function CoverComponents(S, F, G)
2:     for all g – connected components of G do
3:         g.p ← Σ_{f_j ∈ F_g} c_j − |S_g|
4:     while ∃g : g.p < 0 do
5:         g_m ← arg min_g {g.p}
6:         g_M ← arg max_g {g.p}
7:         f ← arg min_{f_j} {c_j | f_j ∈ g_M}
8:         F ← (F \ {f}) ∪ arg max_{f_j} {c_j | f_j ∈ g_m, f_j ∉ F}
9:         Update g.p, g'.p
10:    return F
```

to node v found by Dijkstra on G_b, and $v.p$ the potential of v; there is one Dijkstra execution for each FindPair call.

An assignment is optimal if $\nexists\{s_i, f_j\} \in E'_f \setminus E'$, such that adding (s_i, f_j) to E' would yield a better assignment. Notably, each call of FindPair(G_b, s) updates the running assignment as soon as it finds in E' a shortest path sp from customer s to a non-fully occupied facility. Then the assignment is optimal *iff* E'_f contains no other path sp', from s to a non-fully occupied facility, such that $sp'.length < sp.length$ [257].

Line 12 of Algorithm 2 verifies this optimality condition. Once the condition is satisfied and the loop is over, the assignment is defined for a current E', and the flow augmentation phase follows.

Theorem 1. *Let sp be the shortest path from customer to a non-fully occupied facility in E' and*

$$sp.length \leq \min_{i,j}\{s_i.dist + dist(s_i, f_j) - s_i.p\}. \tag{8.4}$$

Then sp is the shortest path from customer to a non-fully occupied facility in E'_f.

Proof. The Dijkstra's algorithm call in Line 8 of Algorithm 2 adjusts edge weights by *node potentials* to remove any negative cycles created by flow augmentation. The original weight of an edge (v_1, v_2) is $w(v_1, v_2) = dist(v_1, v_2)$, while its *reduced weight* is

$$w_r(v_1, v_2) = dist(v_1, v_2) - v_1.p + v_2.p, \tag{8.5}$$

where $dist(v_1, v_2)$ is the distance between v_1 and v_2 on G. The length of any path in G_b found by Dijkstra is calculated as the sum of reduced weights. Since $\forall v\ v.p \geq 0$, Equation (8.4) implies that

$$sp.length \leq \min_{i,j}\{s_i.dist + dist(s_i, f_j) - s_i.p + f_j.p\}. \tag{8.6}$$

Due to Equation (8.5), Equation (8.6) means that

$$sp.length \leq \min_{i,j}\{s_i.dist + w_r(s_i, f_j)\} \tag{8.7}$$

Now, assume another path sp' from *customer* to a non-fully occupied facility exists that is shorter than sp and includes at least one edge $(s', f') \in E'_f \backslash E'$. Then the length of sp' includes the *reduced weight* of the edge (s', f'):

$$sp'.length \geq s'.dist + w_r(s', f') \tag{8.8}$$

Yet after edge (s', f') is included in G_b, tautologically,

$$s'.dist + w_r(s', f') \geq \min_{i,j}\{s_i.dist + w_r(s_i, f_j)\} \tag{8.9}$$

By Equations (8.8), (8.9), and (8.7), we get a contradiction:

$$sp'.length \geq \min_{i,j}\{s_i.dist + w_r(s_i, f_j)\} \geq sp.length \tag{8.10}$$

□

In contrast, the threshold used by U et al. [257] is:

$$sp.length \leq \min_{i,j}\{s_i.dist + dist(s_i, f_j)\} - \tau'_{max} \tag{8.11}$$

$$\tau'_{max} = \max\{s.p | f.dist \leq \min_{i,j}\{s_i.dist + dist(s_i, f_j)\}\} \tag{8.12}$$

The bound we employ is tighter in case the minimizing s in Equation (8.4) has $s.dist > \min_{i,j}\{s_i.dist + dist(s_i, f_j)\}$ and $s.p > \tau_{max}$. Besides, this τ_{max}-based threshold burdens Algorithm 2 with the overhead of maintaining τ_{max}.

8.6 Analysis of WMA

Theorem 2. *The worst-case complexity of WMA is:*

$$O(m|E| \log n + m^2\ell^2(\log(\ell + m) + k \log \ell)) \tag{8.13}$$

Proof. The matching function of WMA finds a usable facility by iteratively adding new edges to G_b. To that end, it maintains a heap of at most m candidate edges. In the worst case, the heap has to be fully rebuilt at each iteration. If the candidate facility reached by the Dijkstra call on G_b does not satisfy the optimality criterion in Line 12 of Algorithm 2, the loop reiterates. This condition can be violated only if there exists an edge that should be added to E'. As G_b has at most $m\ell$ edges, the Dijkstra result can be invalidated at most $m\ell$ times. Thus, Dijkstra's algorithm is called at most $m\ell$ times. With a heap-based implementation of Dijkstra applied on a sparse connected graph,

the complexity is $O(|E'|\log(m+\ell))$, where $|E'|$ grows iteratively from 0 to $m\ell$. While Dijkstra's algorithm runs on G_b with every FindPair() call, we also run another Dijkstra instance on the graph G for each customer. New edges in E' result from successful executions of that instance, while the heaps for these executions per customer persist across FindPair() calls. This gives an additional $O(m|E|\log n)$ complexity. The combined complexity is:

$$O(m|E|\log n + m^2\ell^2\log(m+\ell))$$

CheckPopular() builds a heap of all reached candidate facilities in $O(\ell\log\ell)$. At each greedy iteration, we check the top value of the heap and update it if needed. In the worst case, we may update the whole heap. In total, we do k greedy steps and return *false* if no set cover is found within the top-k facilities. Then the complexity of the set cover routine per iteration is $O(k\ell(\log\ell+m))$, where m stands for checking whether all customers are covered. The total number of iterations is $m\cdot\ell$, since, in the worst case, we increase the demand of only one customer by 1 in each iteration. Putting it all together, the total worst-case time complexity is:

$$O(m|E|\log n + m^2\ell^2(\log(\ell+m) + k\log\ell))$$

As our experiments document, WMA performs far below this worst-case complexity thanks to its pruning ability. □

Theorem 3. *WMA provides a correct solution if one exists.*

Proof. The main loop in Algorithm 1 terminates when no Δd_i is increased, i.e., when a set cover is found or all uncovered customers reach demand $d_i = \ell$. In both cases, it selects a set of facilities F with cardinality $|F| \leq k$; if $|F| < k$, Algorithm 4 amends it so that $|F| = k$. Algorithm 5 revises F to ensure that all disconnected components of G are allocated sufficient capacity. Let k_g be the minimum number of facilities required to cover all customers S_g within component g, $k_g = \min_{F'}\{|F'| : \sum_{f_j\in F'} c_j \geq |S_g|, F' \in g\}$. A solution to MCFS is feasible if and only if the budget k suffices to allocate to each component g at least k_g facilities, i.e., *iff* $\sum_g k_g \geq k$. Algorithm 5 proceeds towards a state where each component g is allocated a set of top-k_g facilities in terms of capacity values. Therefore, if a solution is feasible, it eventually terminates. Last, the recursive call in Algorithm 1 produces an optimal bipartite assignment from customers to facilities that does not violate any capacity constraint. □

8.7 Experiments

Given the impracticality of approximation algorithms [163], we compare WMA vs. an optimization solver, the *Gurobi Optimizer* [109], and three simple

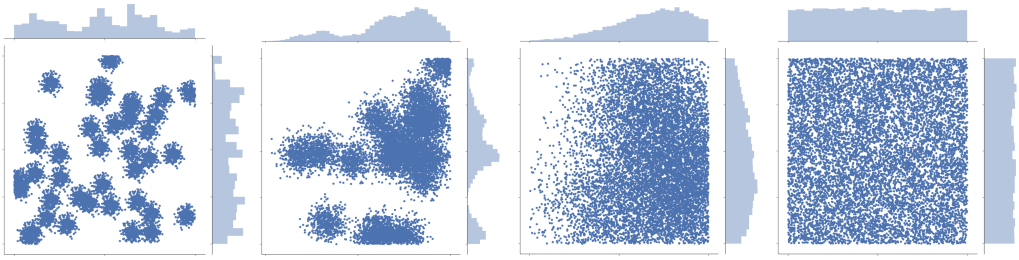


Figure 85: Randomly scattered points used to generate networks.

baselines. Our implementations are in C++. We run all experiments on a 2.2 GHz AMD Opteron 6376 machine with 512GB RAM running Ubuntu 14.04.

Baselines

The first baseline follows an approach as in [188]: it divides the input customer set into k buckets and assigns each bucket to the candidate facility node closest to the bucket's centroid. We form buckets containing $\lceil m/k \rceil$ consecutive customers using the spatial order defined by a Hilbert space-filling curve [136]. We denote this baseline as *Hilbert*. The second baseline is a BRNN-based method that iteratively selects k nodes, calculating NLRs at each step; it then runs SIA to produce a final assignment from customers to selected facilities and obtain the objective value. The third baseline is a simplified version of WMA, *WMA Naïve*. Instead of using an exact bipartite matching, WMA Naïve uses a greedy procedure to satisfy customer demands: in each iteration, it processes customers in a randomly generated order and assigns each customer to its closest d_i candidate facilities that have not yet reached their capacities.

Datasets

We use synthetic and real-world networks. Our real-world data are road networks in Aalborg, Riga, Copenhagen, and Las Vegas, obtained from OpenStreetMap[1]. Table 83 provides statistics. We report objective values and distances in meters.

We create synthetic graphs by placing points on a $10^3 \times 10^3$ square. We use two distributions, uniform and clustered. In the clustered case, we place cluster centers uniformly at random. We then assign an equal number of points to each cluster, and form a Gaussian distribution for each cluster with the center as mean and $\sigma^2 = \frac{1}{\text{number of clusters}}$. We connect pairs of points with an edge if they are closer than $\alpha \frac{1}{\sqrt{n}}$, where α is a tunable *density* parameter and

[1] `https://www.openstreetmap.org/`

	Aalborg	Riga New	Copenhagen	Las Vegas
Nodes	50,961	287,927	282,826	425,759
Edges	55,748	322,109	322,349	508,522
Avg degree	2.2	2.2	2.2	2.4
Max degree	7	29	10	21
Avg edge length	30.2	28.7	32.6	50.4

Table 83: Real-world data sets.

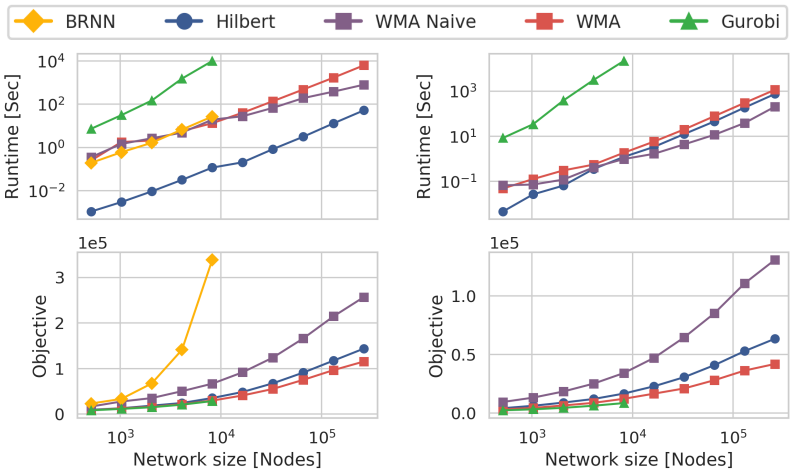
n is the network size in nodes. We connect cluster centers to each other in a clique and assign edge weights equal to Euclidean distances. Figure 85 presents examples of such distributions for 10^4 points given 40, 20, and 5 clusters, and a uniform distribution. On synthetic networks, we select customer locations uniformly at random. A solution is feasible only if there is enough total capacity to serve all customers, i.e., $\sum_{j=1}^{k} c_j \geq m$; in the uniform case, $c \geq \lceil \frac{m}{k} \rceil$, while an *occupancy* value, defined as $o = \frac{m}{c \cdot k} \leq 1$, indicates how close we are to full capacity.

Experiments with Uniform Synthetic Data

We first evaluate performance on uniform data when varying the graph size. We set $F_p = V$, meaning that a facility can be placed on any node in a graph. We present results for Gurobi for instances where it completed within 24 hours. When it does not complete in 24 hours, we say that it fails.

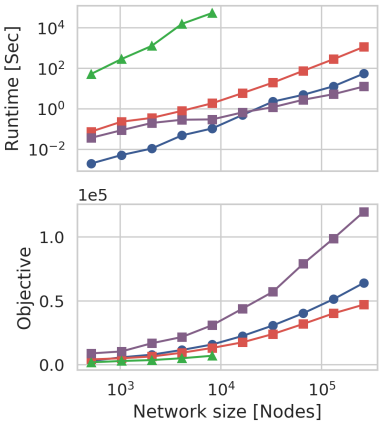
In Figure 86a, we use density $\alpha = 2$, which corresponds to an average of two adjacent edges per node. We randomly assign customers to 10% of all nodes and set $k = 0.1m$; hence, we need to place facilities at 1% of all nodes; we set capacities to $c = 20$, yielding $o = 0.5$, i.e., capacities are twice the minimum required size. BRNN performs significantly worse than others, so we eliminate it from further consideration. The objective values attained by Hilbert, WMA, and Gurobi do not differ significantly, with WMA performing almost as well as Gurobi. This is because this dataset has a simple uniform structure; Hilbert handles it well, even without taking network distances in consideration. However, Hilbert deviates from WMA as data size grows. WMA exhibits a far more scalable runtime trend than Gurobi, which failed on network sizes beyond 8,192 nodes; WMA scales no less gracefully than Hilbert as the data size grows. WMA Naïve has similar runtime to WMA, yet its objective value is more than double that of WMA across the parameter range.

Figure 86b shows results for a similar configuration, but with higher customer and facility density. Here, we set capacities to $c = 4$ and again obtain an occupancy of $o = 0.5$. Results are similar to the previous ones, though the divergence of objectives between Hilbert, WMA, and WMA Naïve is more pronounced. Further, the achieved objective values are smaller for all algorithms

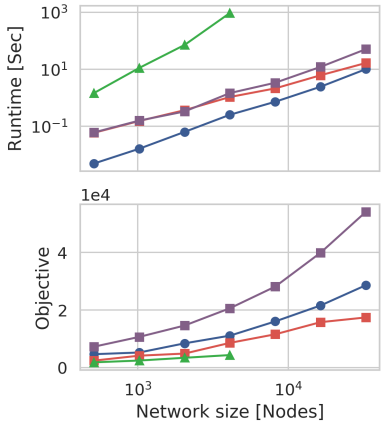


(a) $m = 0.1n$, $k = 0.1m$, $c = 20$, $\alpha = 2$

(b) $m = 0.2n$, $k = 0.5m$, $c = 4$, $\alpha = 2$



(c) $m = 0.1n$, $k = 0.5m$, $c = 10$, $\alpha = 1.2$



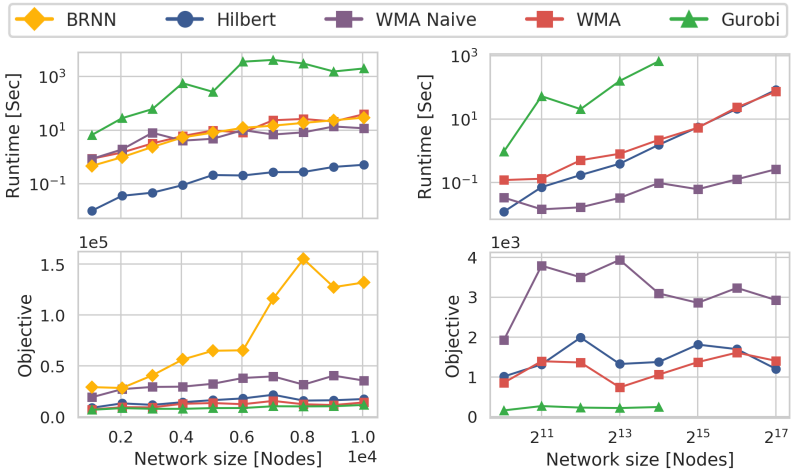
(d) Multicapacities, $c \in [1..10]$

Figure 86: Results on uniform distribution, variable graph size.

due to higher density (the y-axis range has changed). Gurobi's runtime overhead has increased, as the runtime of LP is highly dependent on the number of variables and constraints; the other algorithms are less sensitive to those parameters. Now the runtime of WMA eventually matches that of Hilbert, even while delivering significantly better quality. WMA Naïve is faster than Hilbert on larger networks with higher customer and facilitiy densities, as it eliminates the time-consuming bipartite matching step of WMA.

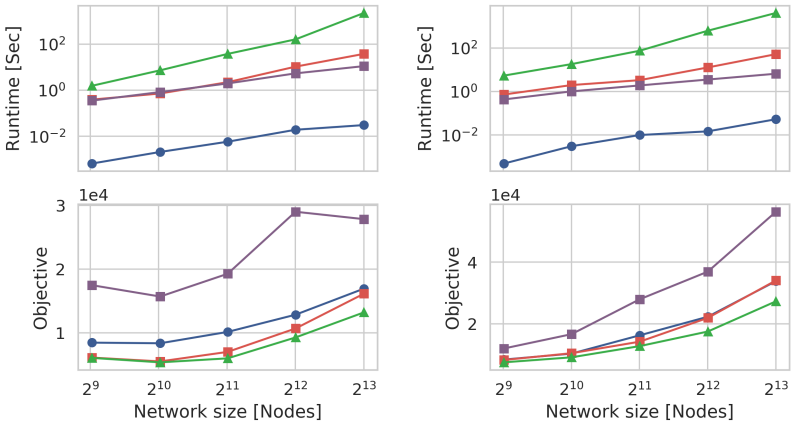
Figure 86c presents a case with a sparser and less connected network, with $\alpha = 1.2$, more similar to real road networks. Customer and facility densities lie between those of the previous two cases, with customers as in Figure 86a and facilities as in Figure 86b. We set $c = 10$, resulting in an occupancy of $o = 0.2$; this makes the problem relatively easier, balancing out the effect of network

sparsity. Even so, the disconnected network structure makes an optimal solution hard to find. Thus, Gurobi's runtime is significantly higher than in the previous case, although the number of decision variables is smaller and the occupancy is looser. WMA also has a higher runtime, and its objective value is closer to that of Hilbert, and similar to that in Figure 86a, where we have half the customers with half the facilities, meaning that the cumulative distances remain relatively stable. Hilbert also has almost the same objective as before, as it considers each component separately, calculating required facilities per component proportionally to the number of customers in the component. On a graph with many small components, this approach quickly leads to good results. As in previous experiments, as the scale increases, WMA Naïve becomes faster than Hilbert.



(a) $m = 0.05n$, $k = 0.1m$, $c = 20$

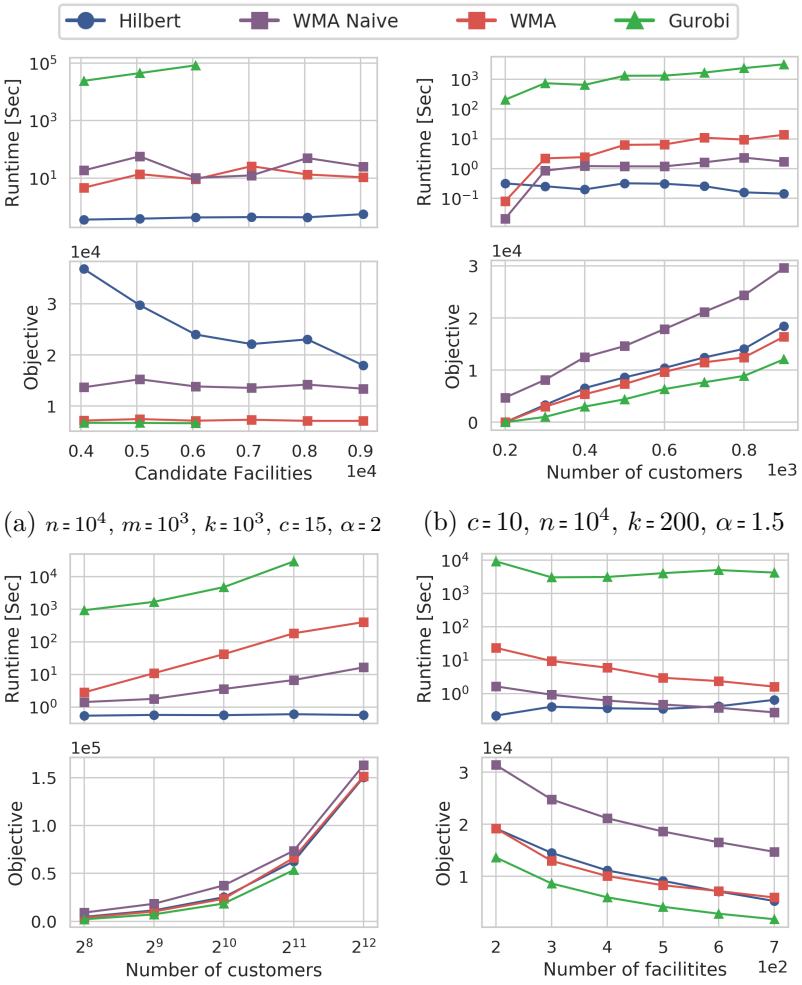
(b) $m = 0.01n$, $k = 0.8m$, $c = 5$



(c) $m = 0.05n$, $k = 0.2m$, $c = 20$

(d) $m = 0.1n$, $k = 0.1m$, $c = 20$

Figure 87: Results on Clustered Distribution vs. size, $\alpha = 2$, 20 clusters in (a,b,c), 5 in (d)



(a) $n = 10^4$, $m = 10^3$, $k = 10^3$, $c = 15$, $\alpha = 2$

(b) $c = 10$, $n = 10^4$, $k = 200$, $\alpha = 1.5$

(c) $n = 10^4$, $k = 128$, $o = 0.1$, $\alpha = 1.5$

(d) $n = 10^4$, $m = 10^3$, $\alpha = 1.5$

Figure 88: Results on Clustered Distribution, 20 clusters. Variable ℓ, m, and k.

We also experiment with nonuniform capacities. Figure 86d shows the results with settings like those for Figure 86c, except that now each node is assigned a uniformly random capacity in the range 1 to 10. Hilbert selects locations first, as if capacities were uniform, and then assigns customers to facilities according to nonuniform capacities using bipartite matching. We observe a similar trend: WMA steadily outperforms Hilbert and WMA Naïve, while Gurobi struggles in terms of runtime. As the problem becomes harder, the gap between the optimal solution provided by Gurobi, and that provided by our heuristic slightly increases in comparison to Figure 86c. The runtime of WMA Naïve is now higher than those of Hilbert and WMA, as it becomes harder for its greedy heuristic to find a set cover when facilities have irregular tight capacities.

Experiments with Clustered Synthetic Data

We now turn to clustered synthetic data. Here, the α parameter no longer corresponds to the average number of adjacent edges per node, as distances between nodes depend on the standard deviations of Gaussian distributions. We tune this deviation so that clusters cover the plane.

Figure 87 shows results for variable network size settings. These results highlight the advantage of WMA further, as the differences between network and geometric distances become more pronounced with clustered data. Hilbert fails to spot good facility locations, as those depend on the network structure. WMA Naïve stands as an outlier with significantly worse results. In terms of runtime, WMA exhibits similar trends as with uniform distributions.

Figures 87a, 87b, and 87c present experiments with highly clustered points. Figure 87a has more customers and relaxed capacity constraints. WMA provides a good tradeoff between effectiveness and efficiency, with both objective and runtime in-between Hilbert and Gurobi. In this experiment we include BRNN, observing that it also underperforms with clustered data; thus, we again omit it from subsequent figures. Figure 87b depicts results for a smaller occupancy and a smaller capacity. WMA performs more similar to Hilbert, though still outperforming it. Figure 87c shows a different low-occupancy setting. WMA and Hilbert yield smoother curves, showing a clear trend. Yet, the problem becomes more challenging for WMA as size grows.

Figure 87d shows results for a case with 5 clusters, coming closer to a uniform distribution, and occupancy $o = 0.5$. Here the clustering-based approaches perform well, with Hilbert becoming almost as good as WMA.

Now we consider the effects of varying the major problem parameters other than network size with clustered data.

Variable number of candidate facility locations

On a clustered graph of size $n = 10^4$, we randomly pick F_p, varying its size from 40% to 100% of all nodes. Figure 88a presents our results, using dense customer distribution and high capacity. Gurobi failed for F_p sizes above 60% of all nodes. Hilbert is sensitive to the size of F_p due to its clustering nature. In contrast, both WMA variants show stable runtime and objective, with the regular WMA achieving objective values very close to those of Gurobi. This indicates that WMA finds good alternatives in case some nodes are not candidate facilities, while Hilbert falters.

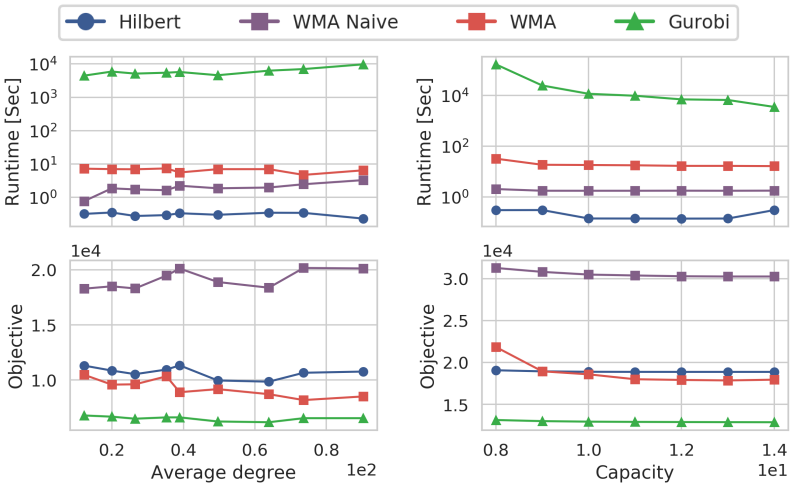
Tuning Customers and Facilities

Figures 88b and 88d present our results when varying the numbers of customers and facilities, respectively. The objective increases as the number of customers grows, but drops as the number of facilities grows, other parameters being equal. Remarkably, the runtimes of the WMA variants drop with

increasing facilities as well, as they perform fewer iterations. Figure 88c scales up the amount of customers, also allowing for multiple customers per node, with occupancy of $o = 0.1$. WMA slightly outperforms Hilbert, and both are very close to Gurobi in terms of objective. WMA Naïve shows worse results. Gurobi fails for large numbers of customers.

Effect of Graph Density α

We now study the effect of graph density α with 5-cluster data. Figure 89a shows the results. As α affects the average degree, the x-axis shows the measured average degree instead of α, resulting in non-equal parameter gaps. The objective improves for WMA with larger degree, coming closer to the optimal solution by Gurobi and outperforming Hilbert and WMA Naïve. WMA finds better locations as optimal facilities become available within fewer hops, thereby simplifying the set cover sub-problem. Gurobi is surprisingly stable, showing that a network with no throughput constraints on edges is resistant to intermediate density increase.



(a) $n = 10^4$, $m = 1000$, $k = 300$ (b) $n = 10^4$, $m = 1000$, $k = 100$

Figure 89: Effect of Density ($c=10$), Capacity c ($\alpha = 1.5$).

Effect of Capacity c

Last, we vary capacity values as Figure 89b shows. The capacity has little effect on the result quality, except in the challenging case of very small capacity, where the occupancy is high. This is reasonable: once a good matching is achieved for some capacity, letting capacity grow further does not improve the solution. Remarkably, While other algorithm have stable runtime, Gurobi gains in efficiency as capacity grows, rendering the optimization easier.

Experiments with Real Data, Uniform Capacities

Now we turn our attention to the performance of WMA on real-world data, using four urban road network data sets of different size. We first examine the uniform capacity case with $F_p = V$. We distributed 512 customers randomly in each city network, and tasked the algorithms with placing 51 facilities. We could only obtain results for WMA and Hilbert, as Gurobi did not terminate on such data within one week due to the large number of candidate facility locations.

Table 84 presents quality and runtime results. WMA achieves a solution that is around 30% better than the most competitive Hilbert basline on all cities except Las Vegas. Las Vegas has a regular grid-like road network structure (see Figure 81a), rendering clustering approaches more effective; thus, we obtain only a 9% improvement.

	BRNN	Hilbert	WMA Naïve	WMA
Aalborg	3.51 / 1 min	0.59 / 10 s	0.83 / 5 min	0.41 / 5 min
Riga	6.02 / 6 min	1.30 / 5 min	1.86 / 3.0 h	0.90 / 3.3 h
Copenhagen	4.20 / 6 min	0.93 / 5 min	1.29 / 3.7 h	0.66 / 5.9 h
Las Vegas	3.67 / 6 min	1.16 / 13 min	1.63 / 12.4 h	1.06 / 7.5 h

Table 84: Objective $[\cdot 10^6]$ / Runtime, $m = 512$, $k = 51$, $c = 20$, $l = n$

Further, we test the scalability of WMA on the Aalborg network, for growing number of both customers and facilities, with fixed occupancy $o = 0.5$, $c = 20$, and setting $k = 0.1m$. Figure 810 shows that the advantage of WMA manifests itself as the numbers of facilities and customers grow: its runtime is aligned with that of Hilbert, and it scales well with the problem size, while the quality improves continuously over that of Hilbert. WMA Naïve achieves a worse objective than WMA, although it is competitive in terms of runtime. Interestingly, as both WMA variants struggle to find a feasible set cover with sparse customers and facilities, their runtimes are at their lowest in middle problem sizes. Further, we ran BRNN on this real-world data set in order to reexamine the conclusions reached on synthetic data. The objective of BRNN grows rapidly, indicating its instability on real-world tasks. In addition, BRNN presents the worst runtime behavior, as it has to repeatedly calculate NLR intersections. Last, Gurobi failed in these experiments.

Experiments with Real Data, Nonuniform Capacities

We now consider real-world data with nonuniform capacities and $\ell < n$, which corresponds to the most general case of the MCFS problem. The problem is to select a set of facilities among diverse options, each associated with a capacity derived from real-world constraints. We study two applications: (i)

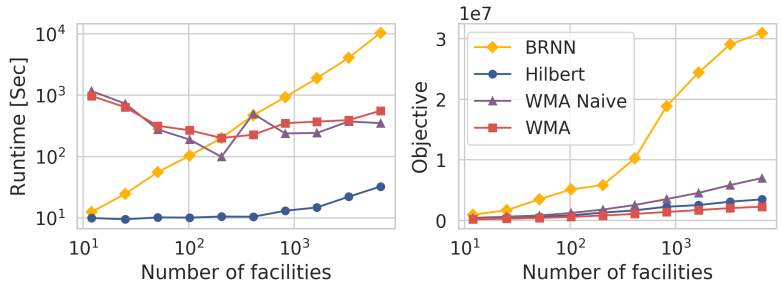


Figure 810: Aalborg experiment, $o = 0.5$, $\ell = n = 50961$

the selection of meeting places for coworkers, and (ii) the selection of bike docking stations.

Coworking

This trend allows independent professionals to share a working environment [93], saving expenses for office rental while enjoying the advantages of the structure and community of working with others [195]. In addition, coworking spaces enable group meet-ups and other temporary activities. Cafés and restaurants provide affordable coworking options, offering part of their spaces during non-rush hours. We let city amenities serve as facilities, while their daily operational hours define their nonuniform capacities. Assuming uniform utilization during these working hours, a number of coworkers need to select coworking facilities out of potential options. We consider this problem on data from two cities: Las Vegas and Copenhagen.

Las Vegas case We use Yelp[2] data to generate a distribution of customers from known facility occupancy, using an existing technique [282]; we divide space to Voronoi cells, and each cell to triangles, as illustrated on the Figure 811.

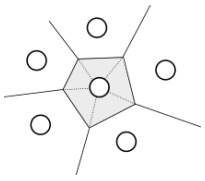


Figure 811: Voronoi cell division.

The number of customers in a triangle is:

$$m_\Delta = O_i \cdot \left(\omega \cdot \frac{O_j}{\sum_j O_j} + (1-\omega) \cdot \frac{Area_\Delta}{Area_{\cup\Delta}} \right)$$

[2]https://www.yelp.com/dataset/

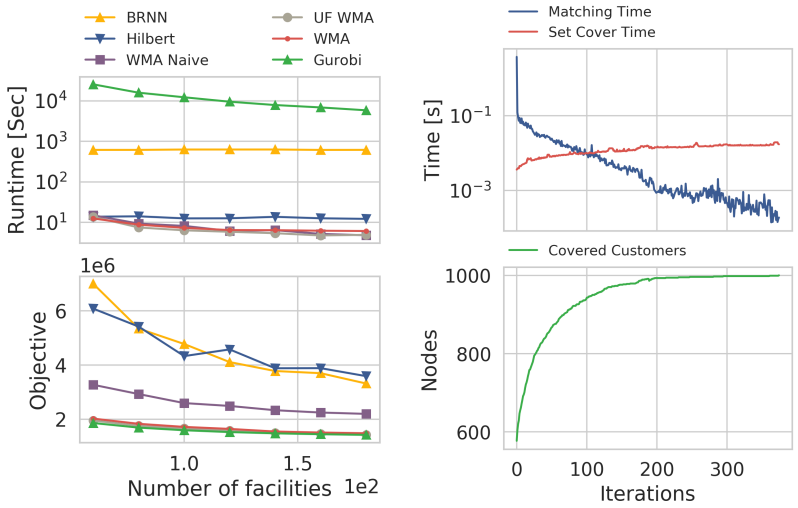
where O_i is the occupancy of the central node, O_j is the occupancy of a neighbor node, $Area_{\Delta}$ is the area of a triangle, $Area_{\cup\Delta}$ is the area of the Voronoi cell, and ω is a parameter set to 0.5 by default [282]. We use user check-ins available from Yelp, considering all restaurants as candidate facility locations, and derive a customer distribution. Instead of using Euclidean Voronoi cells, we adapt the approach to road networks via network distance calculations. We then generate customer numbers proportional to derived values. We place *1,000 customers* at appropriate road network nodes using this method. We downloaded the road map data from OpenStreetMap, and we identified 4089 venues with available operational hours in the Yelp dataset. Figure 81a shows the distribution of customers and facilities in the city center.

Copenhagen case We use data from the "Open Data København" portal[3]. We generate a customer distribution proportional to that of district populations in Copenhagen, and randomly place *200 customers* at road network nodes. We obtained information about cafés and restaurants from OpenStreetMap; 164 venues have operational hours available (the average is 9 hours in both cities), which we use as a proxy for a venue's capacity. Figure 81b shows the distribution of customers and facilities in the city center.

We solve the problem in two ways: (i) the *Direct* solution, whereby WMA accommodates the given nonuniform capacities and proceeds as usual; and (ii) the *Uniform First* (UF) solution, where we first solve the problem as if capacities were uniform using the average capacity, and then reassign customers to facilities using the real nonuniform capacities in a single bipartite matching step. This alternative might represent a better heuristic, in case it detects better locations under uniform capacities, before specializing to the nonuniform ones; this is a conjecture worth investigating.

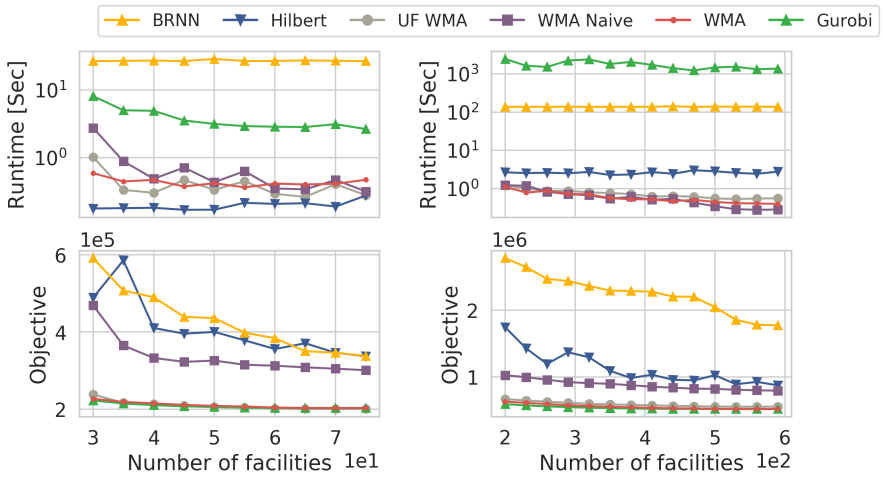
Figures 812a and 813a show our results on the Direct and UF versions of WMA, the optimal solution provided by Gurobi, and the three baselines — Hilbert, BRNN, and WMA Naïve. Since WMA Naïve yields poor quality vs. WMA, we do not include results for its UF variant for the sake of readability. As more facilities can be used to satisfy the given demand, the problem becomes easier. Since we use a small F_p, Gurobi solves the problem in reasonable runtime; that would not be so if we had more candidate facilities or a country-scale network. For both cities, WMA outperforms Gurobi's runtime by several order of magnitude and matches its quality. UF WMA meets the optimal solution as well in most cases. The accuracy of Hilbert improves with increasing number of facilities, replicating the trend observed with synthetic data (Figure 88d). WMA Naïve shows a better objective than Hilbert, as also witnessed in Figure 88a: Hilbert cannot adapt to a small F_p, leading to objectives as bad as BRNN; BRNN has even worse runtime than Gurobi in the Copenhagen case.

[3]http://data.kk.dk/



(a) Selection of meeting places (b) WMA Profiling

Figure 812: Las Vegas experiments.



(a) Selection of meeting places (b) Bike docking case study

Figure 813: Copenhagen experiments.

We also report statistics on the operation of WMA for selection of meeting places in the Las Vegas network with $k = 600$. Figure 812b shows 3 quantities: covered customers at the end of each iteration, time for matching, and time for the set-cover operation. The set-cover time is lower than the matching time except for later iterations where reassignment is minimal. Most customers get covered within the first few iterations. The matching time in the first iteration, where WMA performs a matching of all nodes, is one order of magnitude larger

than in subsequent ones, where it just updates nodes affected by increased demands. The growing number of covered nodes shows how WMA explores the network.

Dockless Bike Sharing

In our second use case, a customer can leave a bike at any place after using it, instead of placing it at predefined docking stations. The rapid growth of companies such as Mobike[4], oBike[5], and Ofo[6] illustrates the popularity of this business model. Still, these companies suggest using "preferable" bike docking stations. Periodically, a service gathers dispersed bikes and distributes them to such stations to enhance the ease of access to bikes. We study the case of dockless bike sharing in Copenhagen, using data from the "Open Data København" portal again. We determine the locations of 6,000 bike docking stations and their capacities (shown in Figure 814). We assume that a new bike sharing company may be licensed a subset of available stations. Our task is to select an appropriate set of k bike docking stations (i.e., *facilities*), observing capacities.

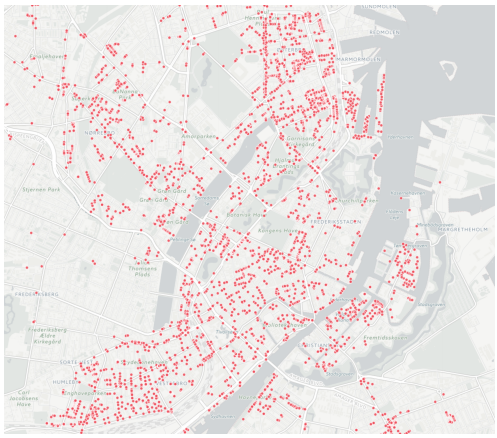


Figure 814: Existing bike docking stations

We generate a distribution of scattered bikes (i.e., *customers*) using aggregate daily bike traffic counter data. A *bike traffic counter* is a point with known coordinates that records the number of bikes passing by in each street direction per hour. Given this information and the default street directions provided by OpenStreetMap, we derive a vector function of *bike flow* per hour, $\vec{g}$. Figure 815 shows the color-encoded magnitude and sign of $\vec{g}$, where the sign indicates the direction of the flow with respect to default street directions. We calculate the divergence $\nabla\vec{g} = \frac{\partial g_x}{\partial x} + \frac{\partial g_y}{\partial y}$ at each network node,

[4]https://mobike.com/
[5]https://www.o.bike/
[6]http://www.ofo.so/

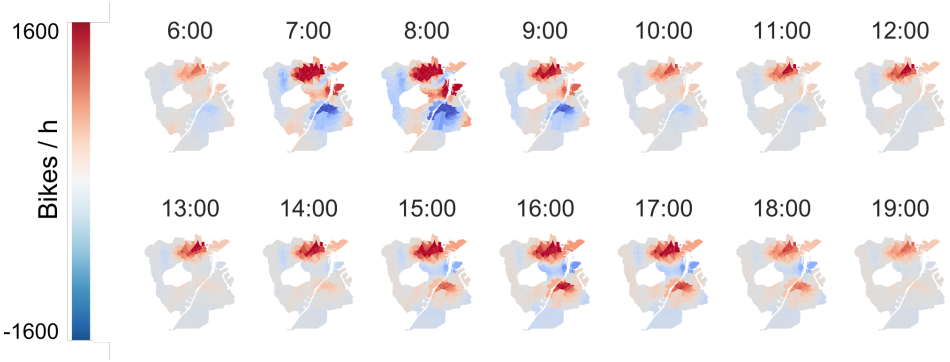


Figure 815: Copenhagen bike traffic

which expresses the number of bikes that get parked at that node during an hour. We repeat this operation for each hour in a day and obtain the *variance* of $\nabla \vec{g}$ across hours at each node, which is a proxy for bike docking demand at that node. Normalizing these variance values, we obtain a probabilistic distribution of bike docking demand across nodes. We place 1000 bikes in the city following this distribution.

Figure 813b presents the results on bike docking station selection. UF WMA fares slightly worse than WMA, while both outperform the baselines and almost match Gurobi.

8.8 Conclusion

We introduced the problem of Multicapacity Facility Selection in a network and presented the first, to our knowledge, algorithm that offers solutions of high quality and scales to large problem instances, the Wide Matching Algorithm (WMA). WMA iteratively builds careful, expanding allocations of customers to usable candidate facilities and terminates when it detects a feasible solution within those allocations. As it can handle both uniform and nonuniform capacities, WMA provides a viable solution for selecting facilities under any capacity constraints. Experiments on synthetic and real-world data demonstrate that WMA is able to solve realistic problem instances; scales gracefully with network size, supply, and demand; outperforms simple baselines in solution quality; and offers competitive quality with respect to the optimal solution.

Chapter 9

Fair Cruising

Ride-hailing systems operate in a two-sided market between passengers and drivers. Such systems manage a fleet of vehicles via two critical operations on the drivers' side: the selection of a route to take when not serving a passenger, or *cruising*, and the assignment of a customer to a driver, or *dispatching*, and thereby affect the market equilibrium. The quality of fleet management has been extensively studied with respect to collective profit on the drivers' side, and satisfaction on the passengers' side, yet less with respect to the *satisfaction* of drivers.

In this chapter, we propose a *maximim* criterion of fleet management quality that expresses the fairness among drivers, with a focus on *cruising* decisions. We find that state-of-the-art cruising solutions based on Reinforcement Learning perform poorly in terms of his fairness objective compared to simple baselines. We adapt these methods based on an enhanced description of the environmental state, and suggest a fairness-oriented combination of cruising and dispatching decisions. Our results show that this adaptation achieves better fairness than state-of-the-art techniques on real-world and synthetic data.

The content of this chapter was submitted to the ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2020 [176], in co-authorship with Leong Hou U and Panagiotis Karras.

9.1 Introduction

Fleet management. *Ride hailing systems* rely on global positioning technologies to provide fine-grained real-time decision-making and service management [132, 263]. Such service manages a large fleet in a manner that tailors the distribution of available vehicles in a city to everyday demand patterns. Recent studies formulate this *fleet management* problem with an optimization objective to maximize total revenue [119, 133, 167, 295]. Two decision-making tasks enter the problem: *dispatching* or *matching* of orders to customers and

positioning or *cruising* of vehicles [133]. Some works consider these two tasks jointly [119, 133, 295]. Yet such joint consideration applies only to environments where matching decisions are centralized.

Cruising. While matching decisions require the consultation of some central authority to avoid conflicts, the *cruising* side of the problem may be addressed as a problem in its own right [49, 167, 202, 209, 219]. For instance, in cities like New York, *street hailing* is the dominant income source for taxi drivers and therefore each driver mainly has a cruising problem to solve [289]; likewise, in ride-hailing services with a *Grab Single* mode, passengers' requests are sent to multiple drivers, hence again each driver needs to adopt a lucrative cruising strategy [70]. The choice of cruising strategy is arguably one of the primary causes of income differentiation among drivers, as more experienced drivers can earn up to four times more than beginners [202].

This introduces the practical problem of *Fair Cruising* (FC) problem, where the goal is to achieve the satisfaction of all drivers as users of a ride-hailing system. This objective is to be achieved by recommending cruising routes to a set of drivers in a manner that maximizes the minimum income among drivers. We find that state-of-the-art profit-maximization strategies do not lead to fair cruising outcomes. We propose algorithms that lead to such a fair outcome, and we show that fairness can be achieved with either minimal concessions or even no concessions at all in terms of total profit.

9.2 Background

Here, we review three areas of research that relate to this work. First, in Section 9.2, we discuss existing works on fair fleet management. In Section 9.2 we examine the state-of-the-art applications of reinforcement learning for profit maximization in fleet management. Last, in Section 9.2 we discuss the general area of fairness in resource allocation.

Fair Fleet Management

Fairness in the context of ride-hailing services has been investigated mainly in the sense of fairness as non-discrimination among *customers* with respect to the service provided [88, 150, 192], the efficiency of recommended routes [211], and pricing [181]. The concept of fairness among *drivers* was first studied in [212]; this study drew attention to the problem of income inequality among drivers, and concluded that the four most significant factors leading to such inequality are the supply-demand ratio, the search distance, the fare, and the speed of delivery. Focusing on a single factor alone, such as operation in a high demand area, or taking long rides, does not guarantee a high income [212]. A similar conclusion arises from a study of income inequality among drivers based on the New York taxi logs [35]; simulations based on the nearest-first and the poorest-first matching strategies indicates that the matching strategy, the

supply-to-demand ratio, and the spatial distribution of trip requests greatly influence driver income inequality. In addition, a cruising strategy by which drivers return to the city center after each trip leads to greater income variance than a strategy by which they wait without moving until the next customer appears; we will include such strategies in our study, among others. However, these works [35, 212] do not propose any method to improve the fairness.

In another direction, Chaudhari et al. [49] consider the problem of *robust* revenue maximization, which is to maximize the minimum revenue of a *single* driver over all possible *trip requests*. A stochastic transition matrix represents possible driver actions, including "going to work" and "going home", which define one's working hours. The same work studies the effect of *dynamic pricing* on the driver's revenue; while a common practice for ride-hailing services is to increase the price during peak hours so as to boost supply, it turns out that such *surge pricing* is misleading; strategic cruising remains the key to maximizing a driver's income.

Some works study the fairness of *dispatching* among drivers. Dai et al. [70] propose poorest-first driver assignment, under the conditions of positive travel cost and fine-grained driver routing. Sühr et al. [243] define fairness as the approximate equality of the revenue of all drivers, taking into consideration waiting times of customers. They propose a Linear Integer Program with a tradeoff parameter that controls inequality levels for drivers and customers. Lesmana et al. [159] provide another tradeoff-based assignment algorithm that juxtaposes the minimum income among drivers with the sum of their incomes. The algorithm repeatedly reassigns the drivers beyond a fairness *threshold* and find an optimal threshold via binary search. However, the aforementioned works do not consider *fairness in cruising*. Idle drivers that are not matched to any customer remain at the same location. As we will see, this strategy leads to poor fairness and also reduces total income.

Ride Hailing as a Markov Decision Process

A common approach to fleet management models the system as a Markov Decision Process (MDP) [202, 219, 253, 283]. Each driver is an agent equipped with a set of possible *actions*. The planning horizon consists of a finite set of discrete time steps. Actors operate in an environment with a set of *states*. A transition from one state of another elicits a *reward*. The goal is to find a *policy* that returns the next action as a function of the *state* of the environment. The optimal policy maximizes the sum of rewards.

The principal approach for policy optimization in Operations Research literature is Dynamic Programming (DP) [219, 253]; the optimization is centered on a *state-value* function Q, which maps states to expected future on-policy rewards. DP requires a perfect model of the environment and therefore suffers from the *curse of dimensionality*: the space of possible actions and states is exponentially large for many real-world problems, especially in a multi-agent

setting. Godfrey et al. [98] apply an *Adaptive Dynamic Programming* (ADP) approach, which achieves scalability by approximating the dynamics of a system by linear functions, to the problem of *stochastic fleet management*; however, this approach is inapplicable in settings where taxi dispatching is allowed to nearby spatial cells [167]. Guestrin et al. [106] represent Multi-agent MDPs as a Dynamic Bayesian network, approximate the value function as a weighted sum of factored linear value functions, which allows for a solution by Linear Programming. The approach, *factored MDP*, has gained popularity due to its scalability in comparison to DP solutions. It has been applied for fair policy optimization, where fairness is expressed as a Nash Equilibrium [285].

Lin et al. [167] show that Deep Reinforcement Learning (DRL) outperforms DP approaches in revenue maximization by *cruising*, which trains policy and value functions using the Advantage-Actor-Critic (A2C) algorithm; unlike the *fully-cooperative* setting, which optimizes central revenue, A2C optimizes the policy for each agent independently; cooperation between agents occurs via an additional constraint, *context*, which forces a driver to accept a destination of lower value. This contextual A2C outperforms other solutions for revenue maximization; we use this algorithm as a baseline in our experiments, and discuss its details in Section 9.5.

Jin et al. [133] propose a *partially* cooperative approach, in which separate modules are responsible for micro and macro management of taxis. The model architecture groups spatial cells into hierarchically ordered regions. Yet this solution cannot address cruising as a stand-alone problem in its own right. It simply considers cruising actions as orders with a negative cost. Contrariwise, we are interested in cruising as a problem in its own right, coupled with *any* dispatching strategy.

Zhou et al. [295] propose a method that embeds a minimization of the Kullback-Leibler (KL) divergence between distributions of customers and cars into the revenue maximization objective. A similar technique is utilized in the *Proximal Policy Optimization* (PPO) algorithm, a state-of-the-art RL algorithm. Like A2C, PPO is a policy gradient method, but with an additional penalization of policy updates at the learning stage. Holler et al. [119] apply PPO to the joint problem of cruising and dispatching. They compare system-centric and driver-centric reward optimization, where the system-centric case implies a *fully-cooperative* setting, and driver-centric is *partially-cooperative*, as in [167]; this algorithm is limited to a few dozens of drivers.

Overall, fully-cooperative Multi-agent RL achieves better cooperation between agents, but scales poorly due to the *credit assignment* problem [167], the problem of a small correlation between an outcome and the actions that lead to that outcome. Fully-cooperative models are limited to a few hundreds of drivers [196], while our dataset indicates more than a thousand active drivers at once.

Pan et al. [202] apply *Inverse Reinforcement Learning* on passenger-seeking habits of drivers to minimize an entropy measure between the observed driver

policy and learned MDP policy. Their examination of habit-based features (number of trips in a cell, average trip distance, traffic, distance to train station/airport) vs. profile-based features (visitation frequency, distance to home, time from start to finish) reveals that experienced drivers gain higher revenue by adapting to traffic conditions.

Fair Resource Allocation

Fairness in resource allocation depends on whether a resource can be divided, and whether actors have individual preferences of their *shares*. The preference an agent a regarding an item i is captured using a *utility function* $u_a(i)$. The fair division problem for a divisible good with individual utility functions is known as the *Cake-Cutting problem* [122]; fair allocation is defined in one of two ways [215]:

- Envy-freeness: each agent prefers their own share
- Proportionality: each agent gets at least an average share

Yet in the case of *indivisible* resources, a fair allocation by the definitions above may not exist [122]. Aziz et al. [15] present polynomial algorithms for deciding on that existence. Alternative definitions include epistemic envy-freeness, minimization of envy-ratio, minimization of a degree of envy [122], and envy-freeness up to one good (EF1) [24].

One relaxation of the proportionality requirement to the case of indivisible goods is the *maximin fair share* [23, 45]: each agent should get a share that is at least as good as the maximum value gained by partitioning the items into n parts and taking the part with the minimum value [24, 94].

Another maximin problem for indivisible goods is the *Santa Claus problem*, where Santa needs to distribute presents so as to make the most unhappy kid happier. Kids have preferences and may get multiple presents. Bansal et al. [19] proposed an LP-based solution. Cheng et al. [59] focus on the restricted case, when a good is available for a subset of agents, each having the same valuation.

Khot et al. [140] assume the utility functions to be *subadditive*: $u_a(i_1+i_2) \leq u_a(i_1) + u_a(i_2)$. As the arising problem is NP-hard, they propose a $(2k1)$-approximation algorithm, polynomial in the number of agents and the number of items. In a manner similar to [159], the algorithm distributes valuable items first, then less significant items, and finally performs reassignment of items for unsatisfied agents according to a fairness threshold estimated by binary search. Fair division is related to job scheduling and load balancing problems, which accept Linear Programming solutions [127]. Job scheduling algorithms are applied to fleet management in [97], albeit without considering fairness objectives.

Maximin objectives also appear in the context of recommendation systems with a minimum required fairness. Zehlike et al. [284] define *group fairness* as the ratio of target group representatives in the top-k ranking, and proposed an algorithm that maximizes the utility of top-k query results, subject to minimum required fairness. Following this work, Lahoti et al. [153] study the problem of maximizing individual fairness. Yet it is not clear how such solutions targeting fairness of representation can be transferred to problems targeting fairness of outcome, such as the problem of cruising recommendation within fleet management, which we examine.

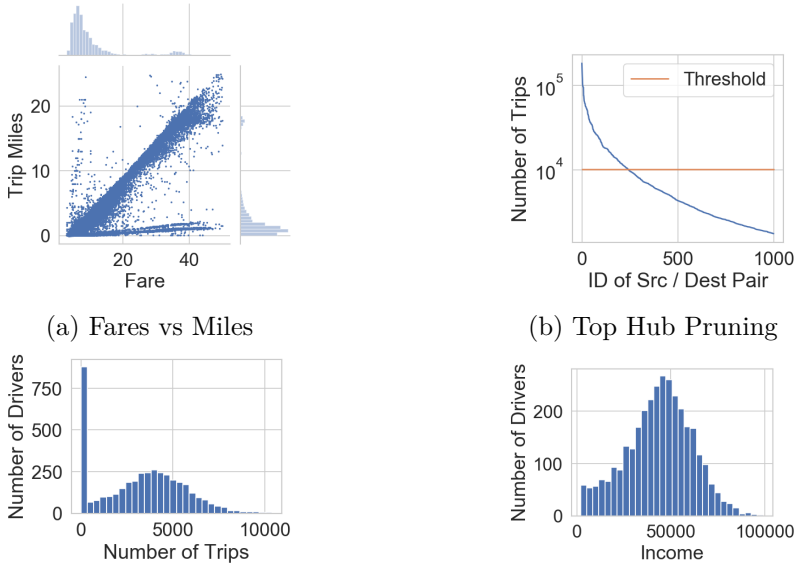
9.3 Problem Statement

We model the setting of our *Fair Cruising* problem via a Multi-agent Markov Decision Process. Consider a planning time horizon (an *episode*) divided into discrete *time periods* $t = \{0 \dots T-1\}$. A city is divided into *spatial cells*. Let G be a directed graph where nodes V represent the cells. Two nodes are adjacent in G if a car can relocate from one corresponding spatial cell to another within one time period. Traffic conditions are out of the scope of this evidential work. We assume the speed to be equal for all the cars at all time periods. We set one *episode* to one day, and one time interval to 15 minutes, resulting in 96 periods per episode. We choose that time interval motivated by the granularity of the real-world dataset we use in the experiments [62].

An *order* is a tuple $c_i =< v_k, v_l, t, \Delta t, p_i >$ with source v_k, destination v_l, time period t of submitting the order, time difference Δt required for a driver to deliver the customer from v_k to v_l, and price p_i for completing the order; v_k may be equal to v_l. Customers are assumed to wait for a car not longer than one time interval (15 minutes). Any unserved order is erased at the beginning of the next time step. A *driver* is characterized in terms of *location* and *availability* to pick up a passenger. At the moment of pick up, the driver receives the corresponding revenue p_i, has location set to v_l, and is considered unavailable during the next Δt time periods.

At each time step t, available drivers are first matched with orders. Each driver that has not been matched follows a *cruising policy* that determines where they relocate at $t+1$. The feasibility of relocation is determined by the network G. We refer to a driver as an *actor*, and denote a *state* of an environment as $\mathbf{s}_t$. A set of *actions* $\{a\}$ is a set of possible destination cells for relocation, according to G. A *policy* is a function $\pi : \mathbf{s} \to \mathbb{P}[a]$ that provides a probabilistic distribution over actions given a state.

The Fair Cruising problem seeks a cruising policy π that maximize the minimum income among drivers. We evaluate policies using a stochastic environment simulator, to be described in Section 9.4.



(a) Fares vs Miles

(b) Top Hub Pruning

(c) Trips per driver

(d) Income per year

Figure 91: Chicago Taxi Trip Data Statistics

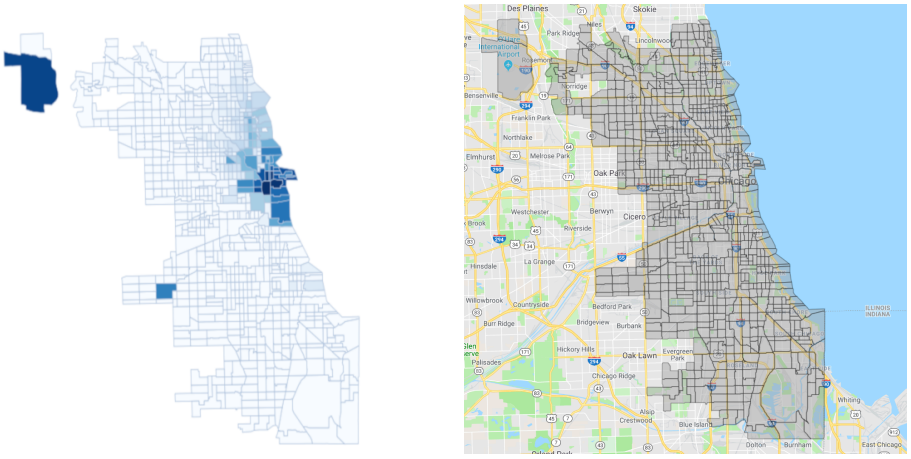
9.4 Benchmark Environment

We use the Taxi logs from Chicago Data Portal [62]. The data includes 183M trips from the years 2013 to 2019. One log entry represents a taxi trip with source, destination, driver id, duration, price, and other trip information. Figure 91c shows the total number of trips per driver over the year 2014; we prune from consideration 800 drivers who made less than 50 trips. The trips of the rest 3321 drivers form a bell-shaped distribution with a mean of 3200 trips per year. Figure 91d depicts the income distribution of drivers. We selected Jan 2014 for simulation experiments; there are 1367 active drivers during that period.

Spatial Network

We build a grid division of the city as a network G using *census tracts* as the basis for that division. Chicago has 801 administrative regions (census tracts) [63], of 35 square miles area in average, as shown in Figure 92b. Due to privacy concerns, the taxi logs' smallest granularity regarding a trip source and a destination is the id of the corresponding census tracts, and timestamps are rounded to multiples of 15 minutes; we set the smallest time interval Δt to 15 minutes.

We define the distance between two adjacent cells as the shortest path length between the nodes in the road network that are the closest to the geometric centroids of the cells. The road network has 474657 nodes. The

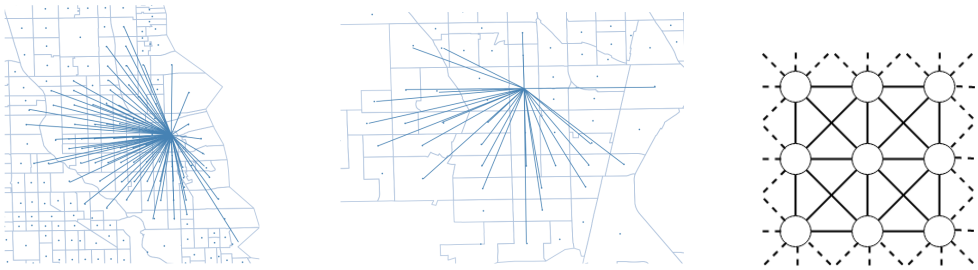


(a) Trip Intensity (b) Census Tracts [63]

Figure 92: Chicago Taxi Trips

average distance between all pairs of cells is 10.0 ± 5.5 miles. According to the log of orders, the average speed is 14.3 ± 8.3 miles/h. As discussed, we assume vehicles are moving with the average speed, therefore any car can travel 3.6 miles in 15 minutes. Based on that, we build the network G, where all cells with a distance up to 3.6 miles are connected with an edge. Figures 93a and 93b present examples of nodes and their incident edges in G. Dots indicate spatial cell centroids. The maximum degree in G is 182. We note that not all the geographically close cells are connected, since some neighboring cells may not have direct road connections. Figure 95b shows the resulting graph G, plotted using the Force Atlas algorithm [25], ignoring the spatial coordinates of nodes. The nodes appear close in the picture to the extent that their neighbors are densely interconnected. Darker nodes indicate a higher degree. We observe that the city's road network forms two dense regions with a sparser part between them. An average trip length that is equal to 13 minutes; 82% of trips do only one hop. Given the average cell area and the average trip length, we derive the average car speed as around 6 miles per hour. We remove records with missing information about source and destination. We consider only the year 2014, as it has the largest number of records (19M).

Figure 91a shows the distribution of the taxi fares versus traveled distance for a sample of 10^5 trips in the logs; we remove toll payments from consideration. Based on the distribution, we remove trips with fare equal to 0$ and larger than 50$, as outliers. Similarly, we disregard trips with the traveled distance of 0 or more than 25 miles. The distribution shows that most trips follow a fare rate of approximately 2$ per mile; 13.7% of trips have a rate of over 16$ per mile (the two bottom lines in the figure), with O'Hare International Airport as source or destination. Less than 0.02% of trips have a rate less than 0.5$ per mile, and appear as a sparse group of points with a



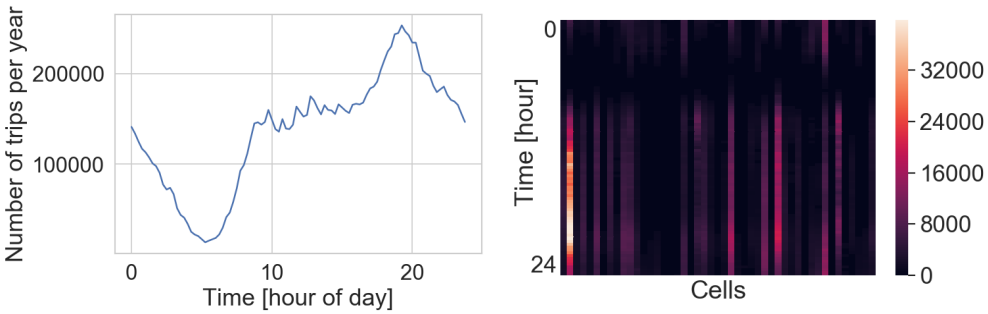
(a) Dense Chicago (b) Sparse Chicago (c) Regular Grid

Figure 93: Examples of connections

large slope. Such trips have higher average speed, which indicates that a taxi followed a highway, while the fare involved a combination of both distance and time. In our simulations, we use the original trip fares.

We also calculate the frequency of trips between census tracts. Out of 642K (ordered) pairs of census tracts, only 55308 have at least one trip (8.6%), 23856 pairs have at least 10 trips per year, and 72.6% of trips are between just 1000 pairs of census tracts. Those 1000 pairs contain only 83 unique census tracts as source or destination. We infer that the city has highly active taxi hubs (*hubs*) represented by a few census tracts, and the rest are passive neighborhoods (*neighs*). All trips are distributed as follows: 3% neigh→hub, 6% hub→neigh, 89% hub→hub, 2% neigh→neigh; most trips are densely concentrated in a single city center.

Figure 94 illustrates the daily activity of cells as sources of trips; in particular, Figure 94a shows how many trips fall in each time period of the day, accumulated over a year, while Figure 94b shows activity per each cell separately, for cells with more than 1000 trips per year. Intra-cell peaks of activity differ across cells, but this difference is small compared to the overall inter-cell difference of activity. Overall, peak hours are between 7 pm and 8 pm.



(a) Total daily trip activity (b) Activity per hour

Figure 94: Daily activity of Demand

Simulator

We train and evaluate algorithms using a taxi trip simulator, which acts on the city network, described in Section 9.4, and a pool of orders collected over a month. One epoch of the simulator is equal to one day. For each time interval of 15 minutes, the simulator samples orders that appeared in the same period of the day. Initial driver locations are set to the locations where each driver took their first order according to real-world order logs. A driver may take an order in the cell where they are, or a neighbor cell. The driver becomes available for new orders at the destination of the passenger after the time length required for delivery. Cars perform one hop per iteration. Delivery within the same cell takes 1 time interval. Network links correspond to 15-minute trips over a road network at the derived average speed of 6 miles per hour.

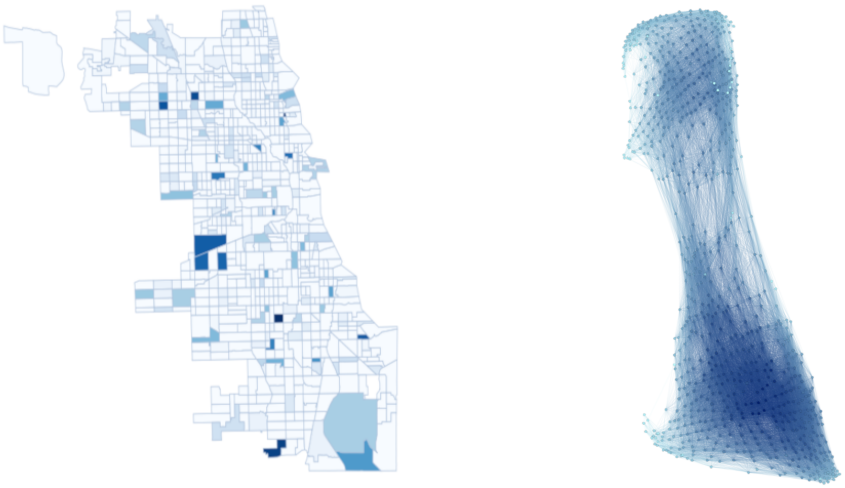
We dispatch drivers as follows. In each node, we first match idle drivers with orders in the same node, then with orders in neighbor nodes, processing nodes in random order. In case the number of drivers exceeds the number of customers, we consider two assignment strategies: we distribute orders among all candidates either uniformly at random (*the random assignment*), or randomly among the k poorest drivers, where k is the number of available orders (*the poorest-first assignment*).

For instance, consider a city with two cells, A and B. A has 2 drivers and 1 order, while B has 1 driver and 1 order. Let $t = 0$. The algorithm picks a cell at random, say B, and matches the driver in B with the order in the same cell. Since there are no idle drivers or unserviced customers, we do not recommend cruising and do not match with neighbor cells. The algorithm proceeds with the next cell, keeping $t = 0$. There, it matches one of the drivers with the order, by either the random or the poorest-first strategy. The other driver does not have a match, so the algorithm recommends a cell to travel to by the cruising policy. The state of the cell contains the *updated* car and order distribution, i.e. $(2, 0)$ for drivers, and $(1, 0)$ for orders. Once the next cell is recommended, the idle driver is set offline (not available) until $t = 1$, and t is incremented.

At the beginning of each time interval, we sample orders randomly from taxi logs of *30 days*, using sampling rate α; $\alpha = 1$ represents the case where the number of sampled orders per time interval is equal to the average number in that time interval over a month. For example, at $t = 0$, the simulator creates n orders, randomly sampled from a set Ω, which contains all orders that appeared in the time window between 00:00 and 00:15 during the first 30 days of 2014, with $n = \alpha \cdot |\Omega|/30$.

9.5 Solutions

We discuss the range of solutions we apply.



(a) Uniform Supply (b) The graph G

Figure 95: Chicago Maps

Naïve Baselines

We use three naïve solutions. *NoPolicy* forces an idle driver to stay in a node until the next assigned order. *Diffusion* selects the next node for an idle driver uniformly at random among all adjacent nodes, including one's current node. *2Hub* sends an idle driver to the nearest hub. We define a node as *a hub* if the number of trips having the node as a source is above a threshold. Figure 91b illustrates the distribution of trip frequencies in year 2014, with a threshold of 10^4, which results in 31 hubs. Figure 92a shows spatial cells coloured according to the number of trips.

cA2C and cA2Cm

Actor-Critic is a family of gradient descent methods used in Reinforcement Learning. Regular Actor-Critic architecture uses a probabilistic policy that maps a state of the environment to a probability distribution over actions:

$$\pi : \mathbf{s} \rightarrow \mathbb{P}[a] \, \forall a$$

The policy is usually parameterized by a neural network with weights θ updated according to the policy gradient

$$\theta \leftarrow \theta + \nabla_\theta J(\theta)$$

which is equal to

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau [\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{s}_t, a_t) A(\mathbf{s}_t, a_t)] \tag{9.1}$$

Here, $\mathbb{E}_\tau$ is an expectation over possible *trajectories* τ, i.e. possible sequence of actions following the probabilistic policy π_θ. $A(\mathbf{s}, a)$ is an *advantage* function showing how much better a is than an average action given state $\mathbf{s}$. The advantage depends on the reward from the environment r_{t+1}, and a *value function* $V(\mathbf{s})$ that shows the future profitability of the state $\mathbf{s}$:

$$A(\mathbf{s}, a) = r_{t+1} + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t) \tag{9.2}$$

γ is a discount factor that limits the planning time horizon. We set $\gamma = 0.9$ at all times.

The value function $V(\mathbf{s}_t, a_t)$ is also parametrized by a neural network with weights θ', trained by minimizing a loss function derived from the Bellman Equation:

$$L(\theta') = \left(V(\mathbf{s}_t) - \sum_a \pi(\mathbf{s}_t, a)(r_{t+1} + \gamma V(\mathbf{s}_{t+1})) \right)^2 \tag{9.3}$$

The method's name comes from an abstraction of the *actor* that chooses an action based on a parametrized policy, and the *critic* that learns the value function, which is used by the actor to update the parametrization of the policy.

Contextual Advantage-Actor-Critic (cA2C) method, proposed by Lin et al. [167], is a variation of A2C, adapted to the cruising problem by *masking* feasible actions of the policy. The probability of infeasible actions in the output of the actor's policy network is set to 0. An action is feasible if $V(\boldsymbol{s}_{t+1}) > V(\boldsymbol{s}_t)$. For total revenue maximization, this helps to eliminate redundant travels, since swapping cars between cells does not have any effect on the objective. However, that is not the case with a fairness objective, where individual income matters, hence swapping cars may be beneficial, if, for example, a poorer car relocates to a more profitable cell. Lin et al. [167] use a 3-layer ReLU network architecture for both actor and critic, with sizes of 128, 64, and 32 nodes, and the standard Adam optimization algorithm for network training, with learning rate $1e-3$.

Both the policy network and the state-value network are shared for all agents. The state of an agent is defined by the concatenated vectors of the driver distribution over cells, the order distribution, and the one-hot encoding of a time period and the agent's location. State vector is normalized. Agents located in the same cell are considered homogeneous. This definition of a multi-agent system is different from a fully-cooperative multi-agent system, where all agents have a global objective. cA2C implies that global revenue maximization is achieved by independently maximizing individual revenues of agents, connected only through the context (action masking), and including a driver distribution in the state. Empirically [167], cA2C performs better if the reward from the environment earned by the agents is averaged over all agents within same cell.

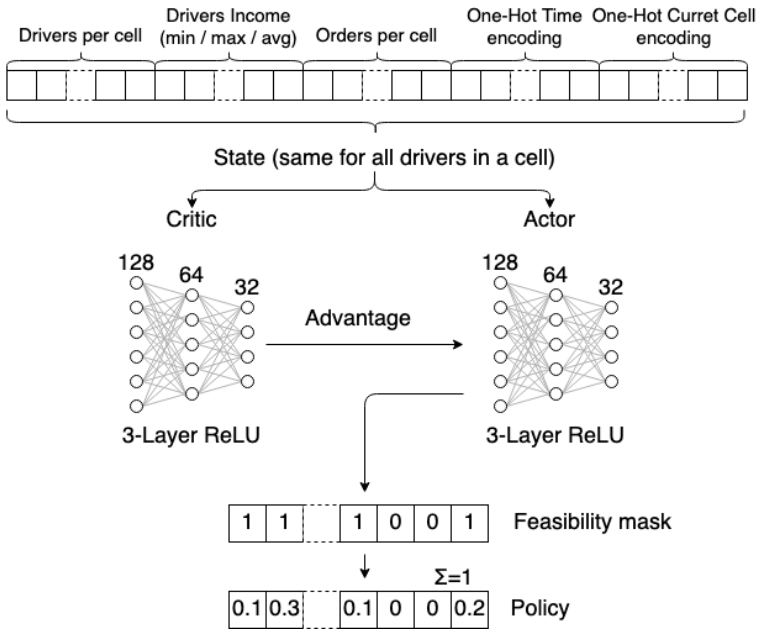


Figure 96: Framework diagram

A technique called experience replay applies. *Experience* of transitions and rewards collected over episodes is stored in a memory, and then a policy is updated by sampling *batches* from the memory. This technique is more stable and sample-efficient than regular A2C, as it removes correlations in the observation sequence and smooths changes in the data distribution [189]. We set a batch size of 3000 in our experiments, as in [167].

Unlike Lin et al. [167], we use a graph representation of the spatial grid. We define an action space of size equal to the maximum degree in G plus 1. The last action corresponds to staying in the same cell. For each node, we define an order over incident edges. A vector of output logits of a policy network contains the ordered neighbours of a node. In case the degree of a node is smaller than the maximum degree in G, extra logits are masked as infeasible.

We adapt cA2C to the fair objective by including the driver revenue into a set of features considered by the policy function. Instead of averaging income per cell, we define reward as the minimum income among drivers in a cell. In addition, we redefine state to include the driver accumulated revenue. Along with the distributions of drivers over the cells and distribution of orders, we include distributions of minimum, maximum and average revenues. All revenue values are normalized by the passed time stamps, and by the largest possible difference between the poorest and richest drivers:

$$income = \frac{income}{t \cdot (p_{max} - \rho)}$$

Here, ρ is a cost of cruising, and p_{max} is the largest price among all orders in the dataset. ρ is estimated by the average fuel consumption [1]. Figure 96 illustrates the framework. We refer to our solution as *cA2Cm*.

Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an advanced policy gradient method, that outperforms A2C on a large collection of single-agent benchmark tasks [231]. PPO has the actor-critic architecture, but, unlike A2C, uses the Kullback-Leibler divergence penalty of policy parameters, improving data efficiency and robustness.

We apply single-agent PPO to the multi-agent setting by recommending cruising for each cell sequentially, in a random order. We update the distribution of cars and orders after processing each cell. As in cA2Cm, a state of an agent consists of the current distribution of cars, orders, and an one-hot encoding of time period and cell id. An action is masked by a set of feasible relocations, yet relocations are not restricted to the cells of a larger state value. The sequential consideration of cells allows to apply PPO implementation out-of-the-box with existing RL frameworks, such as OpenAI Gym [42] and Stable Baselines [117].

9.6 Experiments

We conduct 3 types of experiments:

- Synthetic network and synthetic orders. We test two simple regular grids with different order patterns.
- Real-world network and real-world orders. We build G based on the Chicago road network, and use the Chicago taxi orders log to simulate the environment.
- Real-world network and remapped orders. We use the Chicago road network and taxi logs, but we redistribute the orders so that the overall workload is preserved, but the city has multiple active hubs.

The code of the experimental framework is available online[2].

Synthetic Grid

We start by testing the algorithms in the artificial environments with simple regular patterns. First, we generate a regular 10×10 grid with diagonal connections (Figure 93c), and set $T = 20$ time periods per episode. The grid

[1] `https://www.fueleconomy.gov/`

[2] `https://github.com/allogn/fair-taxi-cruising`

Algorithm	Minimal Income	Total Income [$\cdot 10^4$]	ORR
Regular grid			
NoPolicy	0 ± 0	1.76 ± 0.01	0.85
Diffusion	6 ± 2	1.85 ± 0.01	0.90
2Hub	0 ± 0	1.46 ± 0.02	0.75
cA2C	1 ± 1	1.73 ± 0.03	0.88
cA2Cm	0 ± 0	1.67 ± 0.02	0.83
PPO	8 ± 2	1.81 ± 0.01	0.88
Airport			
NoPolicy	0 ± 0	0.33 ± 0.02	0.02
Diffusion	0 ± 0	1.20 ± 0.09	0.08
2Hub	11 ± 2	7.12 ± 0.07	0.27
cA2C	0 ± 0	1.58 ± 0.03	0.09
cA2Cm	0 ± 0	1.51 ± 0.04	0.06
PPO	0 ± 0	0.73 ± 0.08	0.09

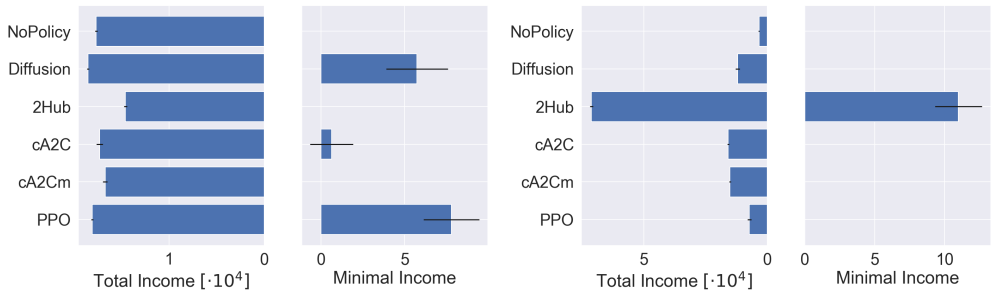
Table 91: Results in synthetic grid

corresponds to the regular hexagonal space division used by Lin et al. [167]. Orders are generated randomly with the density proportional to the proximity to the central node of the grid. The price is equal to the number of hops between a source and a destination. Figure 97a show the result of the experiment. Since each cell has non-zero order frequency, *Diffusion* outperforms other solutions in terms of the total income. *PPO* is the most successful out of the neural network based solution in capturing the simple pattern, and has a slightly larger minimum income than *Diffusion*.

Another result on a simple pattern is presented in Figure 97b. Here, we simulate the existence of two distinctive hubs – a city center and an airport. We use the same regular grid, and we set the positive order rate only for the top-left and the bottom-right corners of the grid. A destination is picked uniformly in random over the grid. The price for the orders originating in the top corner has a multiplier of 10, i.e. 10 times more profitable than from the bottom corner. Although such a setting should illustrate the difference between fair and unfair solutions, we see on the figure that none of the solutions utilize fairness and can achieve non-zero minimal income, except *2Hub*, which strongly dominates in both objectives. Table 91 shows the numerical values for Minimal Income, Total Income and Order Response Rate (ORR) in both experiments with the synthetic data.

Chicago Grid

In this section, we present results for the Chicago-based G and the taxi logs. Trips fares correspond to the logs and are measured in dollars. Figure 99 depicts the Fair and the Total Income objectives. *2Hub* maximizes the total income, while *NoPolicy* shows the worst performance. In terms of the fair

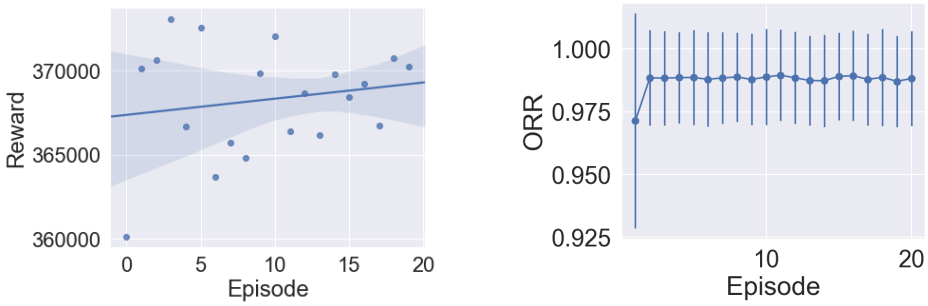


(a) Regular Grid (b) Airport

Figure 97: Results on synthetic grid

income (Figure 99a), *cA2Cm* has a significant advantage over *2Hub*, while all other solvers have zero objective and are omitted from the plot.

Figure 98a shows that despite the advantage of *cA2C* over others, the improvement in the reward r_t throughout training episodes improves only slightly, and with significant variance. This illustrates the difficulty of the training on the fair objective. Figure 98b shows the improvement of ORR over training episodes. We can see, that ORR reaches a high value of 0.974 right after the first training episode. Then, a slight improvement follows after the second episode, and the metric remains static for the rest of the training.



(a) Reward of the environment (b) Order Response Rate

Figure 98: Solution quality per training episode, cA2Cm

Chicago Uniform Grid

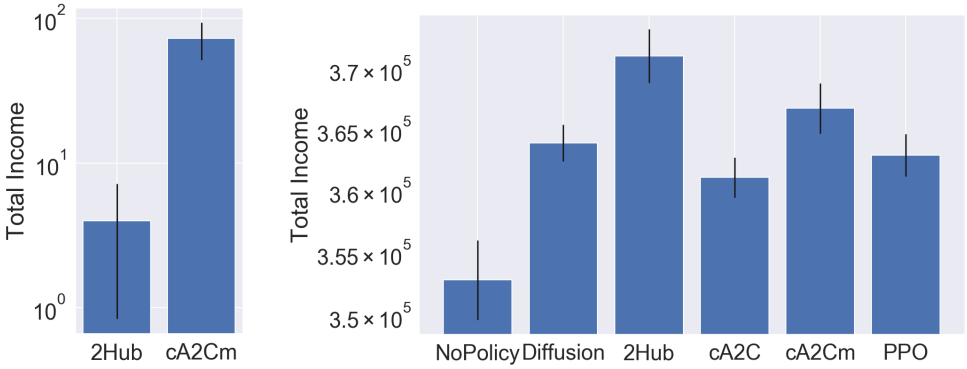
As shown in Figure 92a, hubs of order activity are densely concentrated in the center of Chicago, leaving most of the city inactive. Such setting makes the *2Hub* strategy profitable, and the fair strategy easy-to-learn. In this section, we test a more complex scenario, when a city exposes several spatially distributed hubs. Such setting would require more complex cooperation among agents. We simulate the scenario using the same Chicago road map and taxi

Algorithm	Minimal Income	Total Income [$\cdot 10^6$]	ORR
NoPolicy	0 ± 0	3.04 ± 0.02	0.94
Diffusion	31 ± 65	3.22 ± 0.01	1.00
2Hub	0 ± 0	2.67 ± 0.03	0.83
cA2C	0 ± 0	3.22 ± 0.01	1.00
cA2Cm	0 ± 0	3.22 ± 0.01	0.99
PPO	38 ± 58	3.22 ± 0.01	1.00

Table 92: Chicago Uniform with random matching

logs. However, instead of using the original source and destination of the logs, we remap node ids consistently over the log, so that activity hubs are uniformly distributed around the city. We refer to this case as *Chicago Uniform*. Figure 95a shows the resulting activity pattern.

Table 92 shows Minimal Income, Total Income and ORR of the complete set of drivers in Chicago with random assignment. Due to the high connectivity of the network and an excessive number of drivers, ORR is very close to 1 for all solutions. *2Hub* has the lowest ORR and Total Income, due to the high conflicts between drivers. Figure 910 visualizes the total and minimal income of drivers for all tables with non-zero minimal income.



(a) Fair Income (b) Total Income

Figure 99: Results with Chicago Real Order Distribution

Figure 910b shows the case of the poorest-first assignment. Total income does not change significantly, while Minimal Income improves for all neural-network-based solutions. All experiments hereafter have this type of assignment. in Figure 910d we increase the sampling rate per time period twice. ORR drops, and two other measures increase. *cA2Cm* has the lead in minimal and total income. In contrast, the total income drops for twice *decreased* order sampling rate, as seen on 910e. The Minimal Income in this case is zero for all solvers. Table 93 summarizes the numerical results.

Table 94 summarizes the two cases of non-zero travel cost, where the cost is equal to 0.3 per hop. Figure 910c shows the case of the regular sampling rate, and Figure 910f represents the case where the sampling rate of orders is

Algorithm	Minimal Income	Total Income [$\cdot 10^6$]	ORR
Regular orders			
NoPolicy	0 ± 0	3.04 ± 0.02	0.94
Diffusion	19 ± 31	3.22 ± 0.01	1.00
2Hub	0 ± 0	2.68 ± 0.02	0.82
cA2C	267 ± 161	3.22 ± 0.01	1.00
cA2Cm	392 ± 131	3.20 ± 0.01	0.99
PPO	262 ± 129	3.22 ± 0.01	1.00
Dense orders			
NoPolicy	0 ± 0	6.24 ± 0.05	0.89
Diffusion	567 ± 263	6.80 ± 0.02	0.97
2Hub	0 ± 0	5.55 ± 0.05	0.80
cA2C	902 ± 447	6.82 ± 0.02	0.97
cA2Cm	971 ± 461	6.83 ± 0.02	0.98
PPO	864 ± 486	6.81 ± 0.02	0.97
Sparse orders			
NoPolicy	0 ± 0	1.39 ± 0.01	0.97
Diffusion	0 ± 0	1.41 ± 0.01	0.99
2Hub	0 ± 0	1.19 ± 0.01	0.79
cA2C	0 ± 0	1.41 ± 0.00	0.99
cA2Cm	0 ± 0	1.41 ± 0.00	0.99
PPO	0 ± 0	1.41 ± 0.01	0.99
Sparse drivers			
NoPolicy	2194 ± 1529	1.39 ± 0.02	0.50
Diffusion	4632 ± 2259	1.49 ± 0.01	0.55
2Hub	58 ± 163	1.28 ± 0.03	0.48
cA2C	4332 ± 2216	1.49 ± 0.02	0.56
cA2Cm	5216 ± 1918	1.49 ± 0.01	0.57
PPO	4560 ± 2123	1.48 ± 0.01	0.56

Table 93: Chicago Uniform with poorest-first matching

increased twice. *PPO* shows the best Minimal Income for the regular sampling rate, while *cA2Cm* takes back the lead when the number of orders is increased.

We test solvers under the constraint when a driver can match with customers in the same cell. Table 95 shows the result. With such constraint, Minimal Income is zero for all solutions. In terms of Total Income, *Diffusion* is surprisingly slightly better than others.

In order to test the use case with small ORR, we keep the sampling rate of orders to 1, while decreasing the number of drivers in the city to 10% out of the total amount. Figure 910e show the result. cA2Cm holds the lead in both Minimal Income and Total Income objectives. A notable result is that Minimal Income is much larger in comparison to other use cases.

Algorithm	Minimal Income	Total Income [$\cdot 10^6$]	ORR
Regular orders			
NoPolicy	-29 ± 0	3.01 ± 0.02	0.94
Diffusion	2 ± 54	3.18 ± 0.01	1.00
2Hub	-29 ± 0	2.65 ± 0.02	0.83
cA2C	201 ± 152	3.19 ± 0.01	1.00
cA2Cm	175 ± 117	3.18 ± 0.01	1.00
PPO	290 ± 155	3.19 ± 0.01	1.00
Dense orders			
NoPolicy	-29 ± 0	6.19 ± 0.04	0.89
Diffusion	828 ± 417	6.79 ± 0.02	0.97
2Hub	-29 ± 0	5.55 ± 0.06	0.80
cA2C	843 ± 519	6.77 ± 0.02	0.97
cA2Cm	989 ± 452	6.80 ± 0.03	0.97
PPO	779 ± 409	6.79 ± 0.02	0.97

Table 94: Chicago Uniform with non-zero travel cost

Algorithm	Minimal Income	Total Income [$\cdot 10^6$]	ORR
NoPolicy	0 ± 0	0.49 ± 0.02	0.13
Diffusion	0 ± 0	1.99 ± 0.02	0.64
2Hub	0 ± 0	0.04 ± 0.00	0.02
cA2C	0 ± 0	1.97 ± 0.02	0.63
cA2Cm	0 ± 0	1.98 ± 0.02	0.64
PPO	0 ± 0	1.98 ± 0.02	0.63

Table 95: Chicago Uniform with matching within a single cell

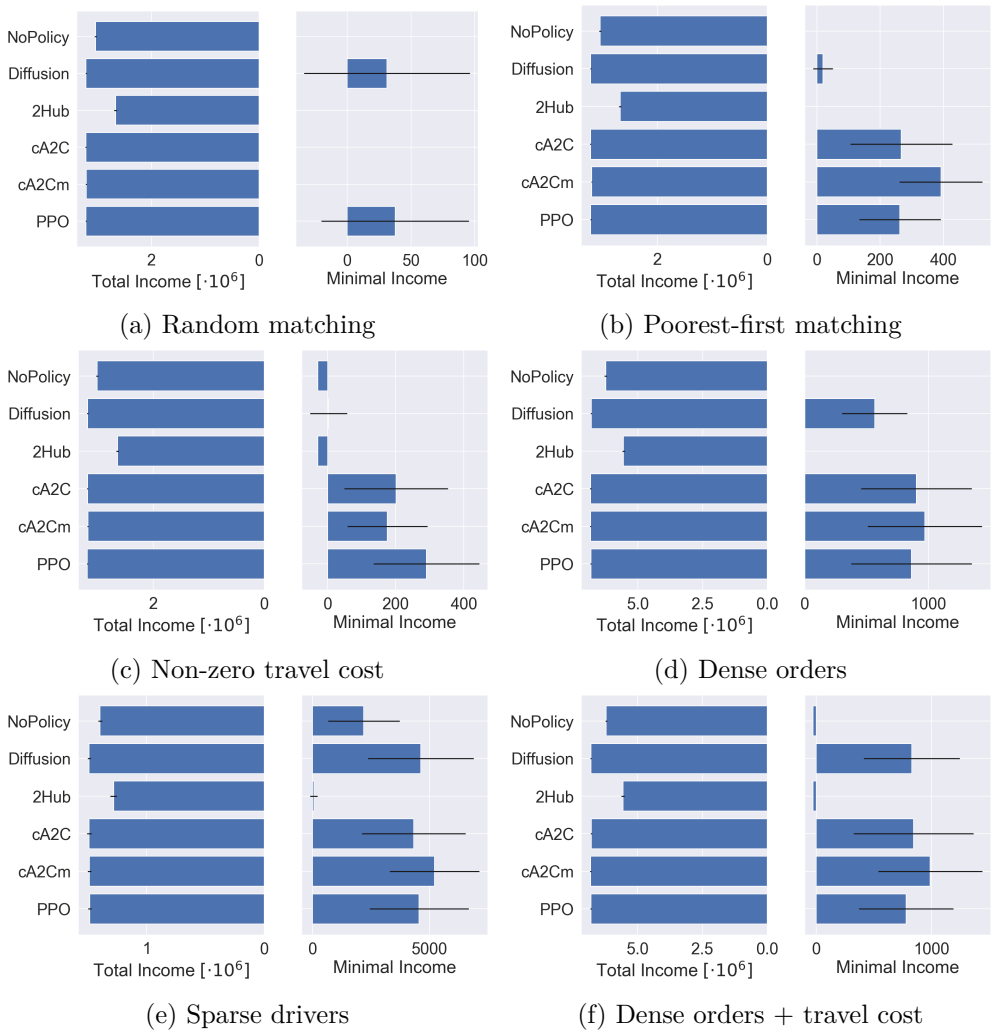
Time-oriented fairness

In this section, we study a business model, when fairness among drivers is achieved by paying a pre-defined salary, or where the salary depends on the total service time, including the idle time. An example of the use case appears in food-delivery companies, such as Delivery Hero[3]. The task of drivers is the same as with ride-hailing companies – a driver requires to pick up an order and deliver to a destination.

Fairness among drivers is a hot topic in this business model as well, and achieved by providing convenient shifts and utilizing the Grab Single mode. While the problem of shift scheduling has been addressed previously in [49], we raise another fairness issue, such as uniformness of a workload among drivers. We define the fairness objective in this case as minimizing the maximum idle time among drivers, and in this section, we address the problem of an optimal cruising for that objective.

We adapt existing NN-based solutions (*cA2C*, *cA2Cm* and *PPO*) by re-defining the environmental reward and state features. Now, a state with

[3] `https://www.deliveryhero.com/`



(a) Random matching

(b) Poorest-first matching

(c) Non-zero travel cost

(d) Dense orders

(e) Sparse drivers

(f) Dense orders + travel cost

Figure 910: Chicago Uniform

expanded features contains minimal, maximal and average non-idle time of drivers, i.e. the time that a driver has *not* been cruising. The state for *cA2C* has not been changed. The reward is set to a minimal non-idle time per cell. Then, the maximization of the revenue corresponds to the minimization of the idle time of drivers.

We compare time-oriented *cA2C*, *cA2Cm*, and *PPO* in the case of a reduced number of drivers, where ORR is middle-range (the same case as the one presented in Figure 910e). Results for time-oriented solvers are shown in Table 96. The naive baselines show zero minimal and average non-idle time. Presented solvers show very close results in terms of non-idle time, with a slightly better average for *PPO*, and minimum for *cA2Cm*. Total Income

Algorithm	Min / Avg Non-Idle Time	Tot. Inc. [$\cdot 10^6$]	ORR
cA2C	24 ± 15 / 28.49 ± 0.31	1.48 ± 0.02	0.56
cA2Cm	$\mathbf{25 \pm 14}$ / 28.48 ± 0.26	1.48 ± 0.02	0.56
PPO	24 ± 12 / 28.54 ± 0.26	1.48 ± 0.02	0.56

Table 96: Time-oriented fairness in case of sparse drivers

hasn't changed significantly in comparison with Figure 910e, dropping from $1.49 \cdot 10^6$ to $1.48 \cdot 10^6$. Minimal Income is zero in all cases.

9.7 Conclusions

We formulated the novel problem of fair cruising in multi-agent taxi fleet management. The objective is to maximize the minimal income among drivers. We studied recent advances in the field and adapted several existing neural-network-based cruising solutions to the needs of this problem, augmenting them with the capacity to perform minimal dispatching operations in favor of the worse-off drivers apart from offering cruising advice. Our experimental study using a real-world dataset, comparing the new solutions and naive baselines, shows that our adaptations reach better fairness objectives when augmented with poorest-first matching, while state-of-the-art solutions for revenue maximization cannot achieve non-zero minimal random matching. Besides, we showed that the same algorithms can be used for an objective of the fair workload under a business model with a pre-defined driver salary.

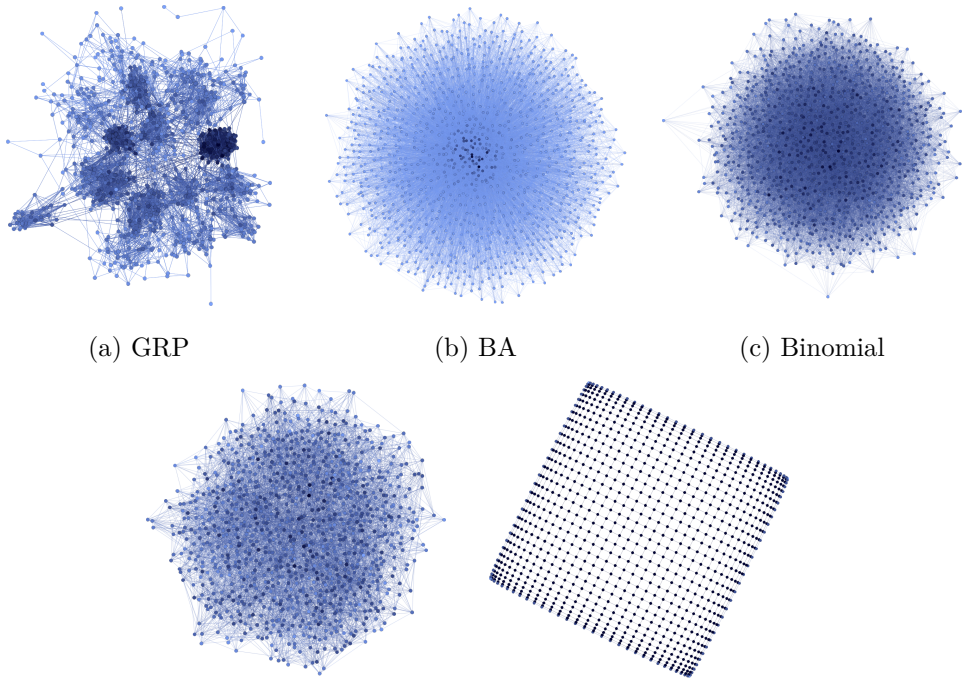
Chapter 10

Network Immunization

Given a network in which a undesirable rumor, disease, or contamination spreads, which set of network nodes should we *block* so as to contain that spread? Past research has proposed several methods to address this *network immunization* (NI) problem, which is to find a set of k nodes, such that the undesirable dissemination is minimized in expectation when they are blocked. As the problem is NP-hard, some algorithms utilize solely features of the network structure in a *preemptive* manner, to others that take into account the specific source of a contamination in a *data-aware* fashion.

This Chapter consists of two parts, Sections 10.1–10.4 and Sections 10.5–10.7. The first part presents an experimental study on NI algorithms and baselines under the independent cascade (IC) diffusion model. We employ a variety of synthetic and real-world networks with diverse graph density, degree distribution, and clustering coefficients, under realistically calculated influence probabilities. We conclude that data-aware approaches based on the construct of *dominator trees* usually perform best; however, in networks with a power-law degree distribution, preemptive approaches utilizing spectral network properties shine out by virtue of their efficiency in identifying central nodes.

In the second part we present a new content analysis workflow which we use to derive the real-world influence probabilities for several diffusion control models; this workflow was also applied in the first part. State-of-the-art algorithms for prescriptive fine-grained diffusion control rely on simple models, most prominently the Independent Cascade (IC) model, rather than on advanced machine learning approaches. The simplicity of such models can be an advantage. Yet, to exploit this advantage, one needs not only well-designed algorithms, but also a powerful model-training framework that yields well-informed models. Unfortunately, much research effort has been devoted to algorithm design, while the development of techniques for informing the underlying model has been largely neglected.



(a) GRP (b) BA (c) Binomial

(d) WS (e) Grid

Figure 101: Generated Graph Examples. Darker color indicates higher degree, normalized per graph.

We rely on a log of user text messages to derive a measure of similarity among those messages, and therefrom calculate the probability that one node influences another. We evaluate our model in terms of its predictive power and apply it to two representative diffusion control problems under the IC model. Our results showcase the capacity of our methods to make correct predictions, and provide the first, to our knowledge, study of diffusion control problems with a real-world probability model.

The content of the first part was published in the 23rd International Conference on Extending Database Technology, EDBT 2019 [171]. The second part was published in the 2019 International Conference on Data Mining Workshop (ICDMW) [172]. Both papers are in co-authorship with Panagiotis Karras. The Chapter contains non-published additional material and extended results, which are marked by an indentation with a coloured bar.

10.1 Introduction to Node Immunization

Real-world networks facilitate the spread of ideas, behaviors, inclinations, or diseases via *diffusion* processes [165]. Oftentimes a diffusion of malicious na-

ture needs to be contained via countermeasures [227]. One such countermeasure is the *blocking* of a subset of network nodes. *Network Immunization* (NI) calls to find an optimal set of nodes to block so as to arrest a diffusion.

Early works on NI were motivated by epidemiology [69, 234], categorizing individuals as Susceptible $\mathbb{S}$, Infected $\mathbb{I}$, or Recovered $\mathbb{R}$. Those who are infected infect their susceptible neighbors with a transition rate β, and become recovered (hence immune) with transition rate γ. In the context of social networks [249], the *Independent Cascade* (IC) model [103] generalizes the SIR model, assigning an independent transition rate β to each edge. Kempe et al. [138] formulated the Influence Maximization (IM) problem under the IC model, where the goal is to select k seed nodes that maximize the expected diffusion spread; since then, the problem has been studied extensively [165, 249].

The NI problem is complementary to the IM problem. Certain notions are useful in both. For example, eigenvalue centrality [234] has been used to guide seed selection in IM. Similarly, Chen et al. [51] employ the first eigenvalue λ as a proxy to the objective of NI problem, scoring nodes by the eigen-drop $\Delta\lambda$ that their removal causes, leading to a succession of techniques aiming to maximize the eigen-drop of immunized nodes [290]. We distinguish two variants on network immunization: *pre-emptive* immunization finds a solution before the epidemic starts; by contrast, *data-aware immunization* tailors the solution to a particular diffusion seed [278]. The state-of-the-art data-aware solution, *Data-Aware Vaccination Algorithm* (DAVA) [291] employs structures called *dominator trees*. Still, the experimental study in [291] is limited to four datasets with synthetic propagation probabilities; it is not clear how the topology of the network influences the algorithms performance. At the same time, recent preemtive immunization methods [234, 247] significantly outperform the baselines used in [291], yet have not been compared to DAVA itself. Thus, to the best of our knowledge, no previous work has studied how data-aware and and preemptive immunization strategies fare under different graph topologies.

In this chapter, we investigate the performance of state-of-the-art data-aware and preemptive NI solutions on a variety of real-world and synthetic network structures with diverse characteristics, and under realistic influence probabilities with the IC model. Our study features the *first*, to our knowledge, application of the most recent algorithm for eigen-drop maximization and a generic spectral method of activity shaping, to NI under the IC model. We demonstrate that data-aware approaches are leading in a majority of configurations, yet preemtive ones stand out under particular settings of graph density, influence probabilities, degree distribution, and clustering coefficients.

10.2 Background

The classic approach to preemptive NI is the NetShield algorithm [51]. NetShield greedily selects a set of nodes S, aiming to maximize its *Shield value*:

$$Sv(S) = \sum_{i \in S} 2\lambda \mathbf{u}(i)^2 - \sum_{i,j \in S} \mathbf{A}(i,j)\mathbf{u}(i)\mathbf{u}(j)$$

where λ and $\mathbf{u}$ are the largest eigenvalue and the corresponding eigenvector of the network's adjacency matrix $\mathbf{A}$. A set S has high Sv if its elements have high eigenscore $\mathbf{u}(i)$ and are not connected to each other (zero $\mathbf{A}(i,j)$). A high eigenscore implies that their removal leads to a significant eigen-drop $\Delta\lambda$. The algorithm has a $O(n|S|^2)$ complexity, where n is the size of a network.

NetShield defines an *epidemic threshold* β' such that any edge transition probability $\beta > \beta'$ would result in a significant portion of the network being contaminated. The algorithm utilizes the fact that the *epidemic threshold* is related to the first eigenvalue of the network adjacency matrix as $\beta' = 1/\lambda$ [264]. Thus, λ expresses the *vulnerability* of the network to an epidemic. Tariq et al. [247] improved upon NetShield by approximating the eigen-drop, relying on the fact that λ can be expressed as the limit trace of the p-exponential adjacency matrix $\mathbf{A}$, which equals the number of p-sized closed walks in the graph, cw_p:

$$\lim_{p \to \inf, p \text{ even}} (trace(\mathbf{A}^p))^{1/p} = \lambda$$

$$trace(\mathbf{A}^p) = cw_p(G)$$

The proposed method greedily selects a set of nodes to block based on their contribution to closed walks, hence to network vulnerability, approximating cw_p by a submodular score function, calculated by partitioning vertices into α equal-size groups by means of a set of hash functions $i = \{1..\gamma\}$. In our experiments, we use $\alpha = 200$ and $i \in \{1..3\}$.

The published version suggested using $p = 6$, yet in communication with the authors we confirmed that $p = 8$ feasibly leads to improved results; we refer to this algorithm as *Walk8*; its complexity is $O(n^2 + \gamma(n + \alpha^3) + nk^2)$.

DAVA [291] accepts the seed set of a network diffusion as input and builds its NI solution around *dominator trees*. A node u *dominates* a node w w.r.t. a seed node s if all paths from s to w pass through u. A *dominator tree* is a tree where each node is dominated by its ancestors. The *benefit* of removing a node is calculated as $\gamma(v) = 1 + \sum_{u \in \text{children of } v} \gamma(u) \cdot p_{vu}$, where p_{vu} stands for the probability that influence propagates along *any* path, approximated via the *most probable* path, from v to u.

DAVA iteratively removes the node of highest benefit and reconstructs the tree. In a DAVA variant, DAVA-fast, the tree is built only once and top-k nodes are selected based on their benefit in one go. A dominator tree is built in $O(E \log N)$ [155].

NetShape [227] immunizes a network via a convex relaxation approach, maximizing the eigen-drop of the network's integrated and symmetrized *Hazard matrix*, a matrix of a continuous integrable transition rate functions $\{\beta(u,v,t)\}_{u,v\in V}$, which indicate the probability that v is influenced by u at time t after u gets infected. The first eigenvalue bounds the expected spread of an infection. We apply NetShape as a heuristic for the IC model, setting the integral of the transition rate $\beta(u,v,t)$ as equal to the influence probability between u and v, and minimize the first eigenvalue by the projected subgradient descent method in the space of possible Hazard matrices *after* immunization, while setting the effect of immunizing u on the integrated hazard matrix element as 0, if u is a seed. The complexity is $O(\frac{1}{\epsilon^2}p_{max}^2 E \ln E)$, where p_{max} is the maximum propagation probability, and ϵ is a parameter affecting a step of subgradient descent.

While our work focuses on immunization under the IC model, more advanced models exist that refer to the similar NI problem under extra conditions. We list some of them.

One of the simplest generalizations of the Data-Aware NI we consider in this work is the *Dynamic Data-Aware NI*, where the budget for immunized nodes is distributed over time. The model implies a possibility to observe a trend of propagation dynamics and postpone the decision. Cui et al [69] expand DAVA to the dynamic case. Wang et al. [262] solve the problem by mixing the Data-Aware model with the compartmental Ising model, one of the simplest models from stochastic graph analysis. Yang et al. [278] proposes another greedy solution based on recursive activation probability estimation. Consider Yang et al. for references on time-dependent breakouts, minimization of budget, NI under the Linear Threshold model, and centrality-based immunization strategies.

A problem closely related to the Dynamic NI is the Stochastic Firefighter problem, presented recently by Tennenholtz et al. [250]. The goal is to derive a sequential vaccination policy, having a budget on immunized nodes per time iteration, rather than per all spreading time. Authors provide a descriptive analysis of the problem under the SIR model and focus on infection growth rate bounds in regular graphs like d-dimensional grids.

Wilder [272] et al. considers a complex model that includes a grouping of population, birth frequency, migration, and uncertain underlying data. Their framework uses submodular optimization to solve a novel multiagent model of disease spread. Other group-based solutions are considered in [226, 291]. [186, 234, 271] consider diffusion shaping in multiplex and data-rich networks, defining models with multi-objective that shape a diffusion over multiple network layers or considering heterogeneous edge and node labels.

Data-Driven immunization, unlike Data-Aware immunization, appears in the literature in the context of the tracking log analysis. [104] works

Graph Type	$\|V\|$ $[\cdot 10^3]$	$\|E\|$ $[\cdot 10^3]$	$deg_{\text{min/avg/med/max}}$	Clust. coeff. cl
Binomial	1.0	14.7	4/15/15/30	0.012
GRP	1.0	14.6	2/30/26/90	0.325
WS	1.0	7.0	18/28/28/44	0.237
BA	1.0	29.4	28/59/40/498	0.103
Grid	1.0	39.7	4/8/8/8	0.000
Stanford	9.9	36.9	0/7/5/555	0.392
Gnutella	62.6	147.9	1/4/2/95	0.007
VK	2.8	40.9	1/29/14/288	0.235

Table 101: Default parameters for graph types

Graph Type	influence prob. W	seed fraction sf	k fraction kf
Binomial	0.2	0.05	0.05
GRP	0.1	0.01	0.05
WS	0.2	0.05	0.05
BA	0.1	0.05	0.05
Grid	0.7	0.05	0.05
Stanford	0.2	0.2	0.2
Gnutella	0.2	0.2	0.2
VK	–	0.2	0.2

Table 102: Default parameters for graph types

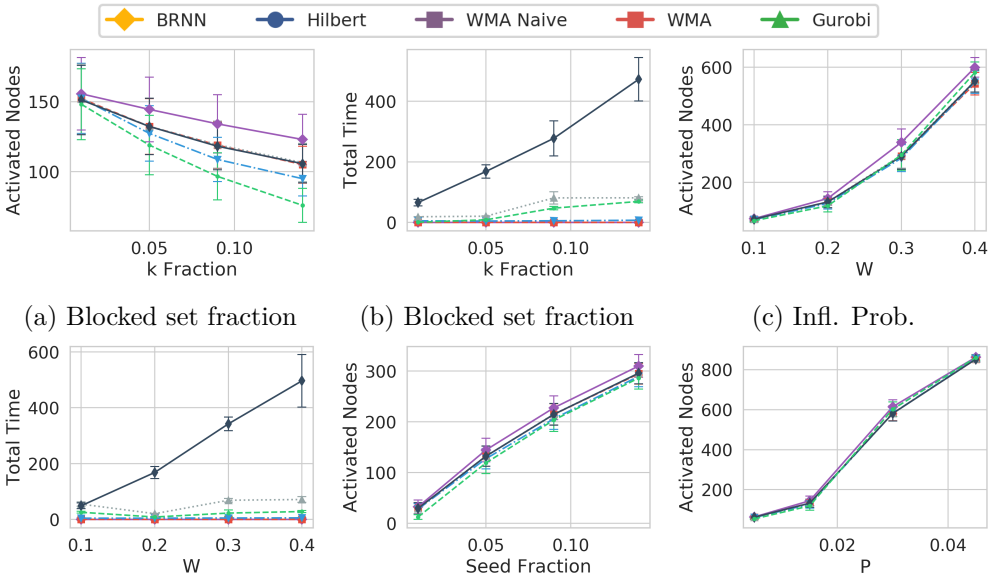
towards inferring the propagation network based on the infection log. Zhang [292] defines Data-Driven NI where immunization strategy is derived solely based on the diffusion log, omitting any diffusion model. They work with so-called contact networks, where nodes represent people, and edges are geographical interactions. The algorithm samples possible cascades of infection propagation and immune nodes that profit in most of the sampled cascades.

10.3 Methodology

Consider a directed graph $G = (V, E)$ with set of nodes V and set of edges E. Each edge is associated with a probability of propagation. By the *independent cascade* model, a diffusion occurs in discrete time steps. In step t_0, a *seed set* $S \subset V$ becomes *activated*. Any node v activated in step t_i attempts to activate each of its inactive neighbors in step t_{i+1}, and succeeds by the probability associated with the edge from v to that neighbor. The process terminates when there are no more newly activated nodes. The *Network Immunization* (NI) problem calls to *block* a select set of k nodes $R \subseteq V \setminus S$ so as to minimize the expected *spread* of activated nodes, by a given seed set S in a graph G.

Graph Type	Other Parameters
Binomial	edge exist. prob. $p = 0.015$
GRP	shape param. $s = 20$, $v = 0.9$, intra-group prob. $p_{in} = 0.4$, inter-group $p_{out} = 0.001$
WS	neighbors in a ring $l = 15$, rewiring prob. $p = 0.3$
BA	prob. of triangle $p = 0.2$, density $m = 15$
Grid	–
Stanford Gnutella VK	–

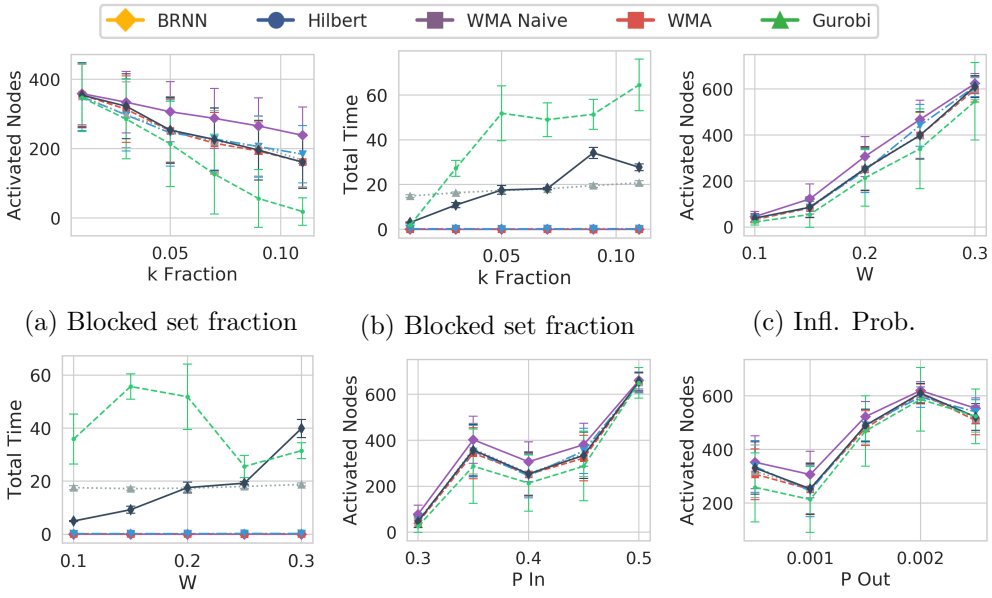
Table 103: Default parameters for graph types



(a) Blocked set fraction (b) Blocked set fraction (c) Infl. Prob.

(d) Infl. Prob. (e) Seed set fraction (f) Edge existence prob.

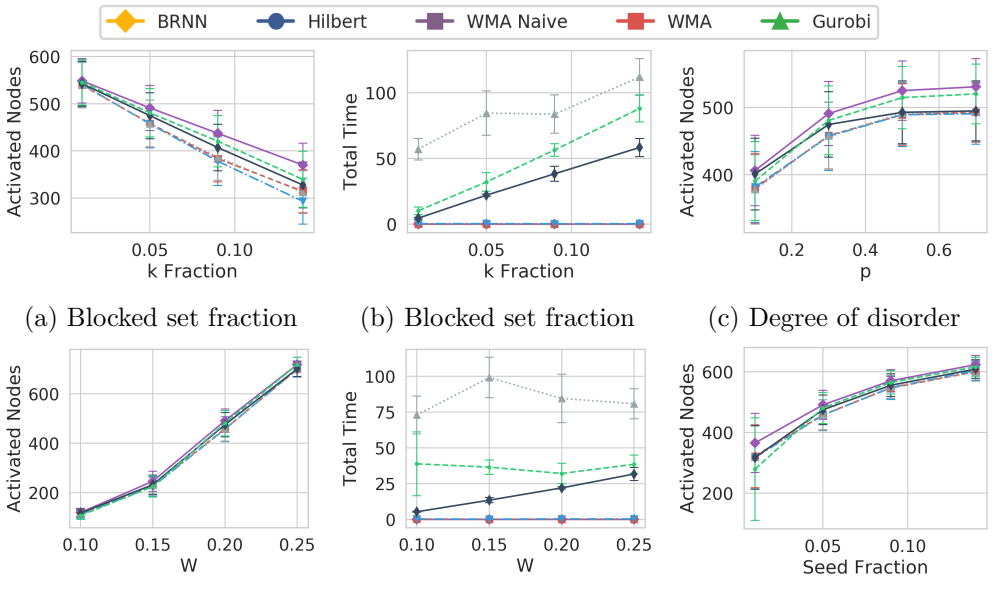
Figure 102: Experimental results on graphs generated by the binomial Erdős-Rényi model



(a) Blocked set fraction (b) Blocked set fraction (c) Infl. Prob.

(d) Infl. Prob. (e) Inter-cluster conn. (f) Intra-cluster conn.

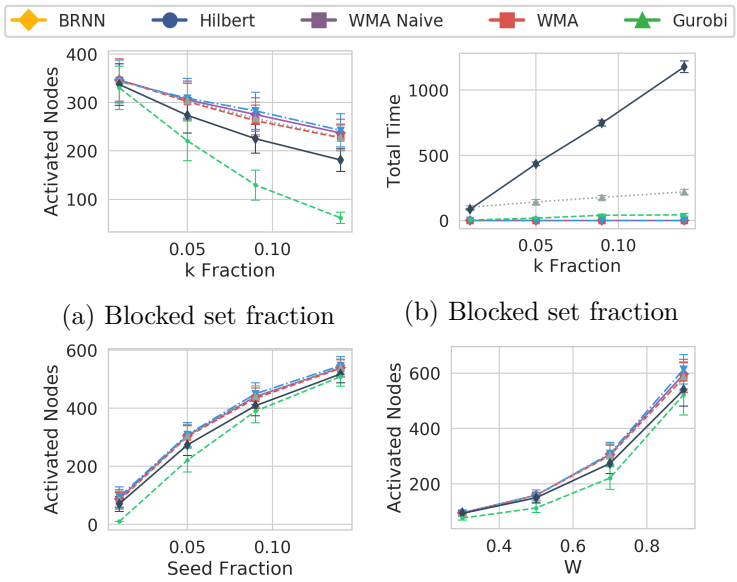
Figure 103: Experimental results on graphs generated by the Gaussian Random Partition generator



(a) Blocked set fraction (b) Blocked set fraction (c) Degree of disorder

(d) Infl. Prob. (e) Infl. Prob. (f) Seed set fraction

Figure 104: Experimental results on Watts Strogatz networks



(a) Blocked set fraction

(b) Blocked set fraction

(c) Seed set fraction

(d) Influence Probabilities

Figure 105: Experimental results on regular grids

Algorithms

We compare six solutions to the NI problem in three categories:

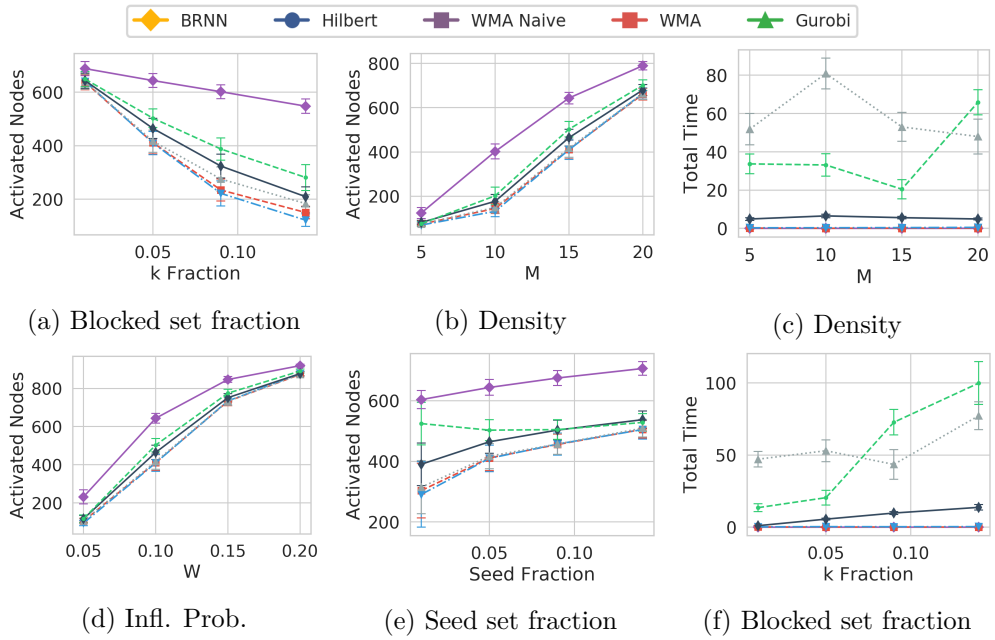
- **Naïve**: *Degree* selects the top-k nodes with highest degree; *Random* selects k nodes uniformly at random.
- **Preemptive**: *NetShield* [51] and *Walk8* [247],
- **Data-Aware**: *NetShape* [227] and *DAVA* [291].

On NetShape, we use the default $\epsilon = 0.2$. As exact spread computation is #P-hard, we estimate spread with any solution via 1000 Monte-Carlo IC simulations. We use the original Matlab code of Walk8. As seeds cannot be blocked, we fetch $k + |S|$ nodes to be blocked with Walk8, ensuring that at least k nodes are blocked. We implemented all other algorithms in Python[1].

Data

We use both synthetic and real data obtained as follows.

[1]Available at https://github.com/allogn/Network-Immunization



(a) Blocked set fraction (b) Density (c) Density

(d) Infl. Prob. (e) Seed set fraction (f) Blocked set fraction

Figure 106: Experimental results on graphs generated by the Barabási-Albert growth model

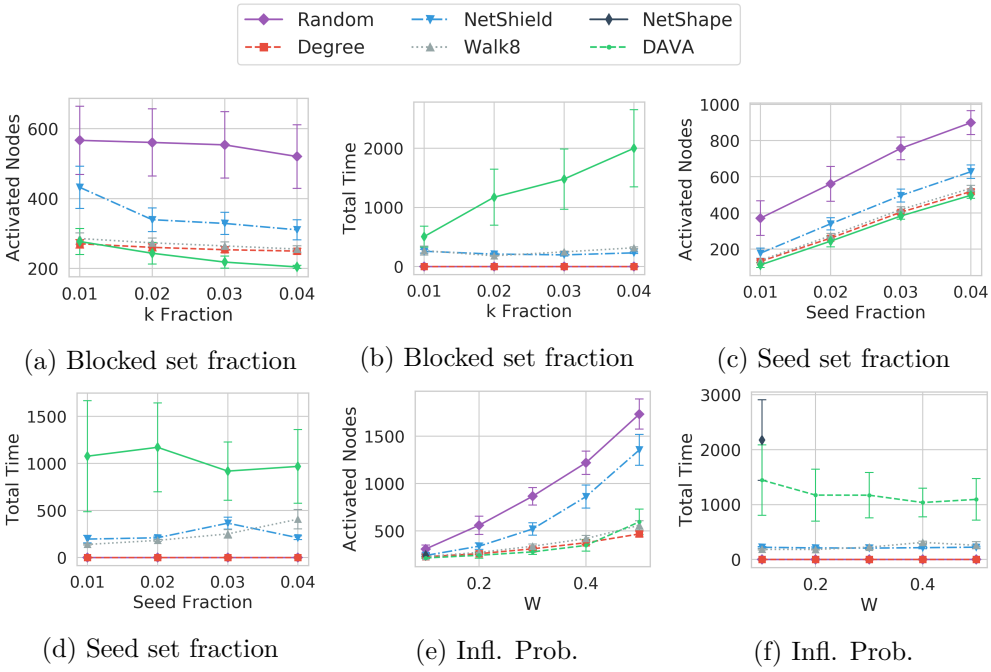
Synthetic Data

We generated graphs of different properties using five models. By the *Erdős-Rényi model*, each edge is present with probability p; generated graphs have a low clustering coefficient and a binomial degree distribution. We refer to this generator as **Binomial**. We render the graph directed by selecting a random direction for each edge with 50% probability.

A **Gaussian Random Partition** (GRP) [41] selects edges as with Erdős-Rényi, but with a prior grouping, where group size follows a Gaussian distribution; it uses a probability value p_{in} for edges across nodes in the same group, and p_{out} otherwise, hence varying intra-group and inter-group density.

Watts Strogatz (WS) networks model self-organizing small-world systems [267], which have small average shortest path length, and are highly clustered, hence susceptible to infectious spread. The generator employs two parameters: l indicates how many nearest neighbors each node is joined with in a ring; p is a probability of edge rewiring, which induces disorder.

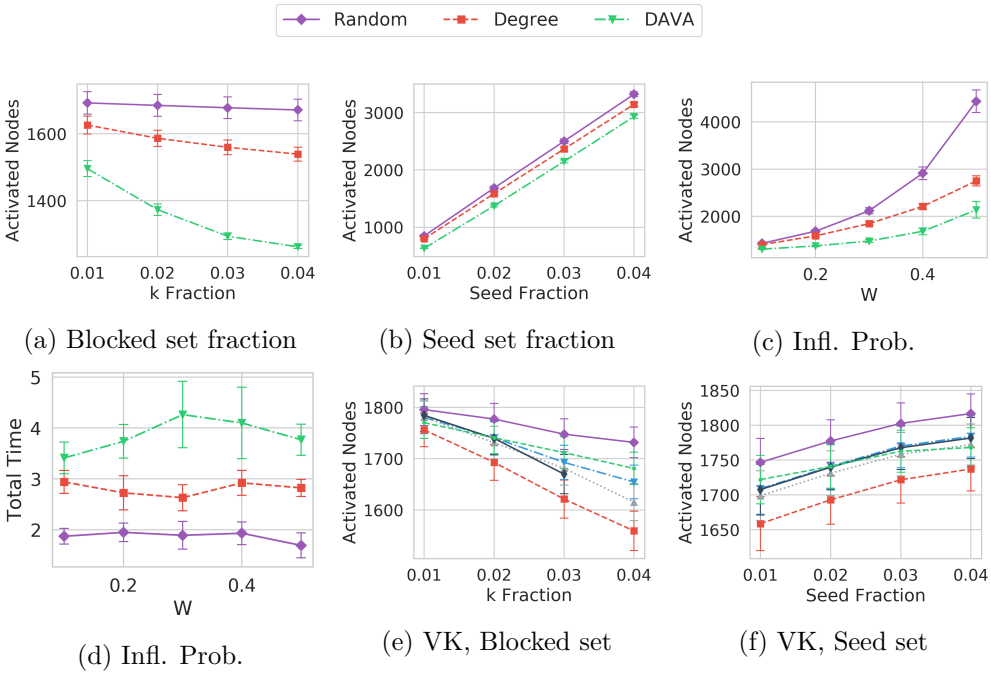
Barabási-Albert (BA) networks have both high clustering coefficients (as GRP and WS graphs) and power-law degree distribution, hence are better imitations of real-world social networks. We use the algorithm of Holme and Kim [120], which extends the original Barabási-Albert model, yet use the BA label as its basis; this algorithm randomly creates m edges for each node in



(a) Blocked set fraction (b) Blocked set fraction (c) Seed set fraction

(d) Seed set fraction (e) Infl. Prob. (f) Infl. Prob.

Figure 107: Experimental results on Stanford network



(a) Blocked set fraction (b) Seed set fraction (c) Infl. Prob.

(d) Infl. Prob. (e) VK, Blocked set (f) VK, Seed set

Figure 108: Experimental results on Gnutella (a-d) and VK (e-f) networks

a graph, and for created edge with a probability p adds an edge to one of its neighbors, thus creating a triangle.

Grid graphs have each node connected to four neighbors on a lattice. With this graph type, we explore the applicability of solutions on spatial graphs such as geosocial contact networks [292].

Tables 101, 102 and 103 list the default parameters for all models, where fractions $sf = |S|/|V|$ and $kf = k/|V|$. Figure 101 shows example graphs. All synthetic graphs have 1000 nodes, as in [264].

Real-World Datasets

We use 3 real-world graphs. Stanford and Gnutella, have been employed in related literature; a third, VK, provides a case of real-world propagation probabilities.

The **Stanford** data consists of pages and hyperlinks in the Stanford University website[2] [291]. The **Gnutella** peer-to-peer file sharing directed network is part of the SNAP dataset [156]. We use the biggest snapshot of 62586 nodes, with a diameter of 11 nodes and a clustering coeficient of 0.0055. It has been used in [227, 278, 291]. **vKontakte**[3] (VK) is a Russia-based social network of more than 500 million users[4]. Its public API allows to download information about public profiles, subscriptions, and posts. We fetch public posts of users to train the IC model.

Parameters

We consider **blocked node set size** k as a fraction of network size [262]. We employ *random* **seed selection** [69, 271, 291]; we pick 10 random seed sets, and show the mean and standard deviation of activated nodes. We choose **influence probabilities** uniformly at random from 0 to a maximum value W. We learn influence probabilities on the VK data using user posts as actions. We download 100 latest posts at the moment of publishing per user, resulting in 21M posts. Most posts are short, hence we can apply the same Natural Language Processing methods as for short messages. After preprocessing, we collected 536,073 non-empty messages belonging to non-isolated nodes in the VK graph, with median length of 11 words, std 187 and max 2977, leaving us with 3% of the original dataset. We define the closeness of actions by comparing the content of text messages, as in [152], to learn vector embeddings of short messages. We define term proximity as $p(w_2|w_1) = \frac{1}{|M|}\frac{c(w_1,w_2)}{c(w_1)}$, where M is a set of all posts with non-zero text content, $c(w)_m$ is the number of messages with w, and $c(w_1, w_2)$ is the number of messages with w_1 and w_2 present together. We learn stemmed term proximities and enrich the term

[2] https://www.cise.ufl.edu/research/sparse/matrices/Gleich/

[3] http://vk.com/

[4] https://en.wikipedia.org/wiki/VKontakte

frequency–inverse document frequency vectors of messages by increasing the probability of any words similar to words present in the message. We consider all message pairs with similarity above the median as similar. Scanning the action log to calculate the influence probability from a node u to any nove v as the ratio of successful reposts of similar messages. Filtering zero-probability edges, we select the largest component of 2.8K nodes and 40.9K edges as our VK network.

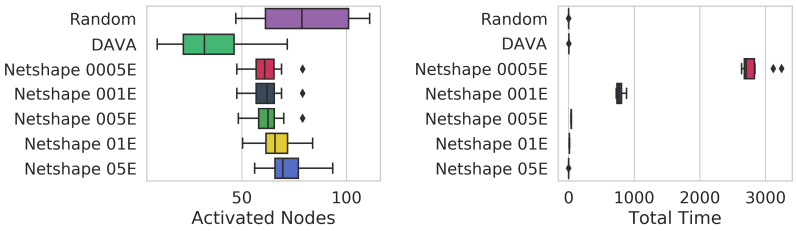
10.4 Experimental Results

Here we present the results of our study. We set a timeout of 1h for all experiments for a single solver instance.

Synthetic Data

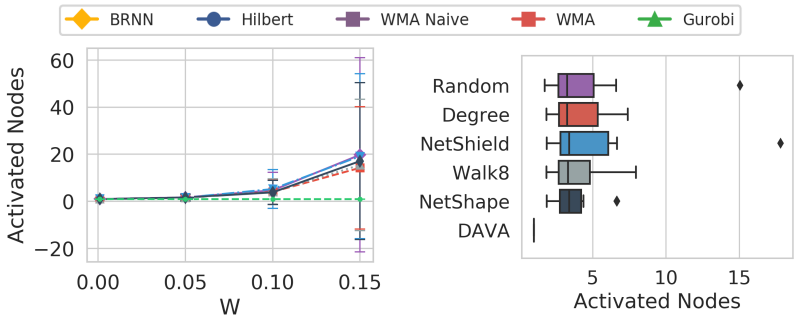
Figure 102 shows results with **Binomial** graphs. As the number of blocked nodes grows, DAVA's advantage of knowing the seeds becomes evident. Surprisingly, NetShield achieves better results than NetShape and Walk8 in this graph type. As the graph has a uniform structure, spectral-based algorithms do not perform well. This uniformity results in performance of algorithms not being dependent on the number of seeds and influence probability W. Still, as Figure 102c shows, with large W DAVA is slightly worse than preemptive approaches. DAVA assumes that the influence probability between two successive dominators in the dominator tree is equal to the probability along the shortest path. When there are many paths between two dominators, this assumption fails, hence the accuracy of the algorithm drops. We observe that NetShape is the least scalable algorithm.

Figure 103 shows results with **GRP** graphs. Again, the gap increases as k grows. DAVA achieves the best results on all parameters, except for the largest p_{out}. We observe that, as the inter-group probability p_{out} grows, DAVA shows slightly worse performance; in other words, as the graph forfeits its clustered structure, DAVA provides less accurate probability estimates.



(a) Activated nodes (b) Runtime

Figure 109: Varying the Netshape ϵ, GRP graphs.



(a) Edge probabilities

(b) Default setting

Figure 1010: Results on GRP graphs with seeds selected by IM.

Figure 109 shows the NetShape performance for varying ϵ. As ϵ drops from 0.1 to 0.005, performance improves slightly, yet the runtime blows up. Still, even with the smallest $\epsilon = 0.005$, the solution quality is worse than that of DAVA.

Figure 1010 presents the behaviour of algorithms with low probabilities and target seed selection. We select only one node of highest degree as a seed, while the compared algorithms select k nodes to block as usual. If $k \geq deg(seed)$, then the optimal solution isolated the single seed. Figure 1010a shows that only DAVA succeeds to block the single seed, finding the optimal solution for any graph. Figure 1010b shows the results for other algorithms. NetShape, the only query-based algorithm, performs slightly better, while Walk8 gain an advantage over NetShield. Surprisingly, Random performs better than Degree.

Figure 104 shows results with **WS** graphs. Here, preemptive algorithms perform significantly better than DAVA, while the difference is accentuated as the number of blocked nodes k grows.

Figure 105 shows results with **Grid** graphs. All algorithms except DAVA fail to isolate seeds. NetShape outperforms other spectral approaches thanks to its data-awareness. Runtimes are similar to those in the Binomial case, with DAVA being sufficiently scalable.

Last, Figure 106 shows results with **BA** graphs; the degree heuristic and NetShield perform best. This result indicates that there are limits to the versatility of DAVA.

Real Data

Figure 107 shows results on the **Stanford** network. We employ the fast DAVA that builds a dominator tree only once so as to scale. Exploring a larger range of parameters than [291] reveals that DAVA performs similarly to the

Degree heuristic, and slightly worse as W grows, due to the scale-free data topology. NetShape and Walk8 could not scale to such size. **Gnutella** has a more random topology than the Stanford network. Running on the 62K-node Gnutella snapshot, only fast-DAVA and baselines terminated within the time limit. Figure 108 shows the results, with DAVA reasserting its advantage. Our **VK** graph has high clustering coefficient and power-law degree distribution. Figure 108 shows that, on this data, DAVA is outperformed by preemptive methods. We deduce that, in real-world social networks, isolating diffusion sources is less critical than immunizing influence hubs.

10.5 Introduction to Diffusion Control

Network diffusion. Phenomena of diffusion in networks involve the spread of information, attitudes, or infections. Some of those phenomena are captured by the *Independent Cascade* (IC) model [103], which is has been applied in problems such as Influence Maximization (IM) [138, 165] and Node Immunization (NI) [291]. The IM problem asks to find a set of k nodes in a network that maximizes the expected spread of a diffusion starting out from these nodes. On the other hand, the NI problem asks for a set of nodes that, if removed, would minimize the expected spread a diffusion emanating from some other source. These problems arise in social network analysis, viral marketing, and epidemiology [69, 234].

IC parameters. The IC model requires a single probability value that describes the influence between two nodes that are connected in the network. Seed nodes are initially *active*. Once any node is activated, it tries to activate (influence) its neighbors, and succeeds with the corresponding edge probability. The assessment and exploitation of diffusion control methods requires an accurate estimation of IC model probability parameters, as these probabilities may emphasize network localities. For example, as Zhang et al. [288] show, in real-world networks, the retweeting probability is negatively correlated to the amount of social circles a user belongs to, i.e. the number of friends that do not know each other. Such effects are hardly captured by synthetic models.

IC training Despite the wide usage of the IC model, the question of its *training* has been poorly covered. Some works train the IC model based on a set of *actions*. For instance, Saito et al. [225] learn model parameters using an expectation-maximization approach, applied on a set of references to a particular topic in a blogging platform. Netrapalli et al. [194] use such approach to learn graph parameters and the graph itself. Goyal et al. [105] suggest a more scalable approach to learn model parameters; they define a propagation probability between two adjacent nodes as $p_{uv} = \frac{A_{v2u}}{A_{\text{total}}}$, where A_{v2u} is the number of alike actions performed by v and then by u, and A_{total} is the total number of actions performed either by v (A_v), or by both users

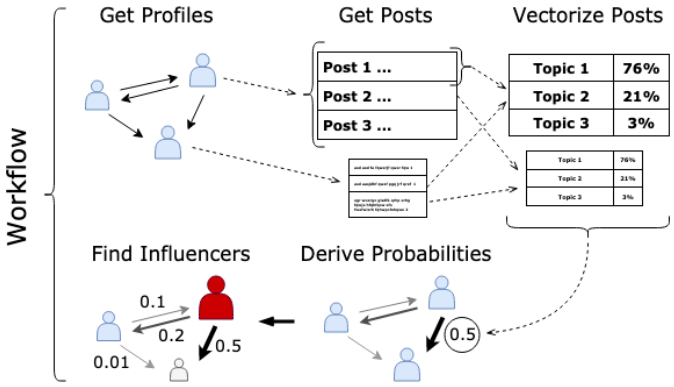


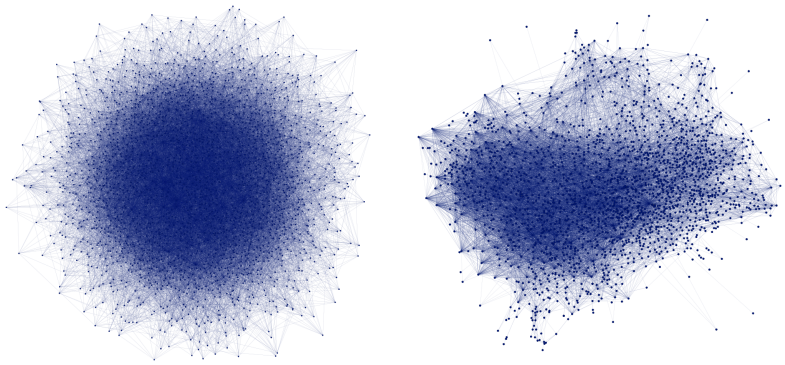
Figure 1011: Framework for extraction influence probabilities

($A_{u|v}$); they apply this solution on a dataset of Flickr accounts, where actions are group subscriptions.

Content-aware IM Other works [21, 124] introduce content-aware variants of the IM problem. Most pertinently, Barbieri et al. [21] apply content analysis to the topic-aware IC model, a generalization of the IC model where the propagated item (e.g., a piece of news) is associated with a *topic vector*. Likewise, each edge is associated with a *vector* of influence probabilities, calculated in terms of actions of its adjacent nodes. The activation probability over an edge depends on its *topic closeness* to thr propagated item, defined in terms of the item's topic vector and common interests of adjacent nodes. Still, the topic-aware IC model assumes topics are the *ultimate cause* of influence across network nodes. This model cannot infer influence due to the *relationship* among nodes. For example, a user subscribed to a news source she trusts will tend to be influenced by that source, regardless of topic. In this case, content analysis is useful for *inferring* influence (i.e., the user posts messages of content similar to that of posts by the trusted source), yet the inferred influence is due to topic-independent factors. To our knowledge, there has been no attempt to train the IC model based on such general inferences from the *content* of exchanged messages, where items belonging to the same cascade may not be identical.

Frequent flow paths. Subbian et al. [242] study the problem of mining flow paths in a graph having frequency above a threshold f; thus, their focus is not to extract probabilities of flow across each edge, but to identify sequences of nodes across which information flows frequently.

Influence prediction Some works use richer models than the basic IC to deliver better *prediction* of influence. Cheng et al. [58] use message and user features to predict the size and shape of cascades by machine learning. Liu et al. [169] use Gibbs sampling to train joint probabilities of influence between users as well as between items, modeled as distinct types of nodes in



(a) BA Graph (b) VK Graph

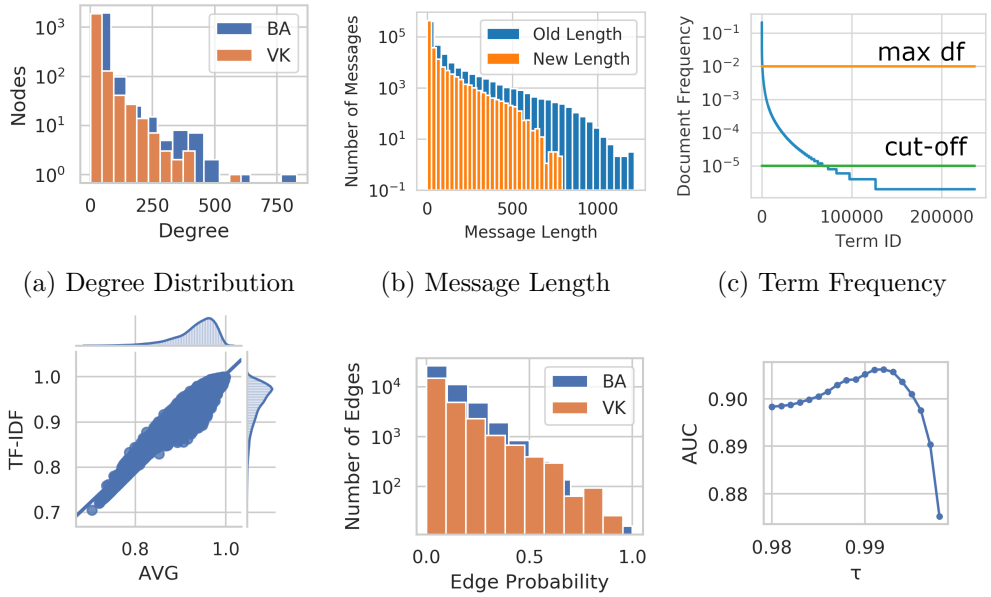
Figure 1012: Synthetic (left) and Real-World (right) Dataset

a heterogeneous network. Wang et al. [266] use an embedding model and apply Maximum Likelihood Estimation for prediction, using the network structure for regularization. However, these prediction models are inapplicable with modern *diffusion control* algorithms, as they do not scale to the computations such algorithms rely upon, and violate those algorithms' assumption of edge-based influence independence.

In this part of the Chapter, we propose an end-to-end framework, that so far has been missing, for the extraction of accurate, general-purpose, topic-independent influence probability values from the real-world content of user text messages. We start out with a topic vector representation of messages, similarly to previous works, yet consider circumstances in which, in a propagation trace, the content of a reposted message has no reference information and is not identical to, but merely *similar to*, the original. We emphasize that we do *not* propose some new model with better prediction power than others; instead, we propose a new learning framework for the *existing*, widely used IC model, so as to enhance the real-world applicability of IC-based diffusion control algorithms, and conduct the first, to our knowledge, experimental study of such algorithms under model parameters trained by real-world interactions.

10.6 Framework

The Independent Cascade (IC) model captures a diffusion in a network. Let $G = (E, V)$ be a directed graph, where nodes correspond to users of a social network and edges correspond to subscriptions. A node can be in active or passive state. Each edge is associated with a probability $p{:}E \to (0, 1)$, which indicates how likely it is that the source influences the target. Information propagates in discrete time stamps t_i. Nodes that are active at t_0 are called seeds. If a node v becomes active at t_i, then for each $(v, u) \in E$ such that u is



(a) Degree Distribution (b) Message Length (c) Term Frequency

(d) Message Similarity (e) Probability Distribution (f) AUC

Figure 1013: Dataset statistics

not yet active, u becomes active at t_{i+1} with probability $p((v,u))$. We train the IC model by assessing similarities on a log of posts, using tokenization and morphological analysis, as follows.

Data Collection

We use the VKontakte (VK)[5] social network. A *profile* in VK represents either a single user, or a community page (a *group*). The VK API allows to query only public data of active profiles. A profile can subscribe to another profile. A mutual subscription is called a *friendship*. We model profiles as nodes in a social graph. Each profile has a *wall* — a blog with posts. A post may contain a text message, pictures, audio, video, or documents. We use text message contents to derive post similarity, and ignore all attachments. A post is labeled by publishing time, author, content, and a history of reposts. A *repost* is a post that refers to another post; a post can refer only to one other post; the referred post appears on the wall of the repost author, along with content by the author. We do not concatenate the texts of all reposts, yet we define all reposts to be similar to original posts by default. Yet, as only 0.2% of all downloaded posts are reposts, we can not use reposts for influence probability learning. Therefore, we calculate message similarity based on content. We query the following information about profiles:

[5]https://vk.com/

- Information about a profile, including state (active or inactive).
- The list of subscribers of a profile.
- The latest 100 posts from the wall of a profile, including a history of reposts; reposts may belong to any network profile, including closed or deleted ones; this limit is due to VK API's constraint on a query to a profile's wall.

We query groups and users independently. We initiate the database with a few random nodes, and continue to query nodes on first-come-first-served basis, whereby new node IDs come from the retrieved lists of friends/subscribers. For those nodes with known lists of friends, we queried the latest posts.

After retrieving several thousands of nodes, we change the approach so as to enhance graph connectivity. First, we query a detailed information about a set of profiles. We then query lists of friends and posts for publicly available profiles with more than 10 connections; we filter those with less than 10 connections to reduce data sparsity. Then we go through a process in which we iteratively collect profile ids by a priority queue, increasing the score of a node v by 1 each time v appears as a friend of a user in the previous set, and by 2 if v has authored any repost from the previous set.

Eventually, we extract a list of nodes, each having a full friend list and wall available and appearing in at least another node's friend list, amounting to $5.6 \cdot 10^5$ nodes and $1.5 \cdot 10^8$ edges; 14% of those are user profiles, while others are group profiles; groups are more likely to set public profiles. We select nodes having at least 5 posts and pick the largest weakly connected component to obtain a graph of 2452 nodes, 28108 edges, and 106,217 posts; the graph, depicted on Figure 1012b, has clustering coefficient [228] 0.121 and mean degree 22; its degree distribution appears in Figure 1013a.

Probability Extraction

We used as tokens morphological word lemmas obtained by MyStem[6] [233]; we ignore unrecognized words, e.g., URLs and smileys, and omit stopwords with the NLTK library [31]. Figure 1013b shows message length distribution before and after preprocessing; in VK there is no character limit.

To calculate the similarity of posts, we need a term similarity measure. We considered two such measures: (i) the Jensen-Shannon divergence of term co-frequency, suggested in [152]; and (ii) a skip-gram model [34] pretrained on Wikipedia, provided by FastText[7]. FastText utilizes word stem information to represent words that are not in the training data. Figure 1014 shows the correlation among these two measures, based on pairwise cosine similarities, by on two measures, among a random sample of terms. We see that there is

[6] `https://tech.yandex.ru/mystem/`
[7] `https://fasttext.cc/`

no much correlation. Table 104 shows examples of some of the most similar (translated) terms according to FastText. Based on the observation of such examples, we opted for FastText.

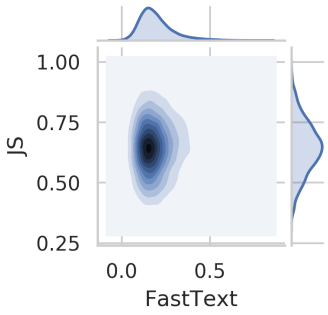


Figure 1014: Term sim.

Term 1	Term 2	Similarity
mess up	curl up	0.7704
pour	knead	0.7066
economic	sociological	0.6649
button	shirt	0.6308
fanatic	aggressive	0.6113

Table 104: FastText sim. examples

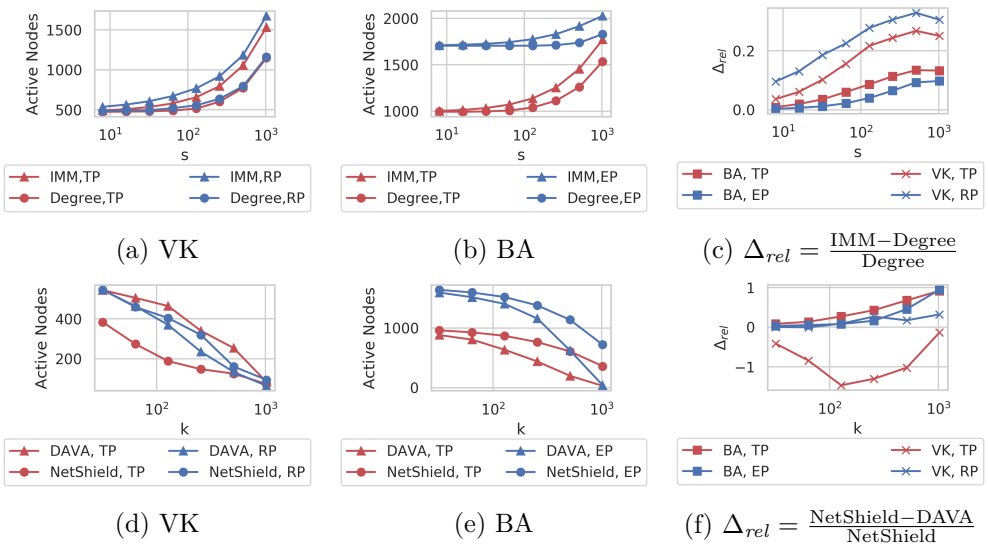
Next, we consider two ways to calculate post similarity: (i) *AVG* [241], the cosine similarity among the averages of term vectors in each post; (ii) *enriched TF-IDF* [152], which is tailored for embeddings of short messages, as it amplifies the entry for a term w in the tf-idf vector of a message m using all terms w' present in m, weighted by similarity to w:

$$\text{tf-idf}_{w,m} = 1 - \Pi_{w'}(1 - \text{tf}_{w',m} \cdot \text{idf}_{w'} \cdot p(w|w'))$$

where $p(w|w')$ is the cosine similarity between term embeddings, given by FastText. We applied a minimum document frequency (cut-off) of 0.001%, and a maximum document frequency of 1%, yielding $6.6 \cdot 10^4$ terms. Figure 1013c presents term frequencies. The minimum frequency filters words that appear too rarely to influence message proximity, while the maximum frequency indicates words that appear too often, and are therefore unlikely to reflect specific context. Figure 1013d juxtaposes the two post similarity approaches on a random sample of 1000 posts; in this case, as opposed to the case of term similarity in Figure 1014, we observe a high correlation. We opt for enriched TF-IDF as the more refined approach.

Eventually, we calculate edge probabilities by scanning the log of posts. We define influence probability as $p_{uv} = \frac{A_{v2u}}{A_v}$, where A_{v2u} is the number of posts by user v that are *similar* to an *earlier* post by u. We consider a pair of posts A from u, B from v, as *similar* only if (i) A satisfies a *content similarity threshold* τ with respect to B, i.e., $\cos(\text{tf-idf}_A, \text{tf-idf}_B) \geq \tau$, (ii) A precedes B by at most one month, and (iii) no earlier message B_2 from v satisfies τ with respect to A, and no later post A_2 from u satisfies τ with respect to B; in other words, we assume a user is influenced by a post *only once*. Figure 1013e shows the resulting probability distribution.

For each post m and each neighbor of its author, we consider the existence of a similar post m' on the neighbor's wall as a *positive* instance of propagation. We use a cutoff threshold $\theta \in (0, 1)$ to determine our probabilistic predictions of propagation. Scanning the log of posts, as in [105], we derive True Positive



(a) VK (b) BA (c) $\Delta_{rel} = \frac{\text{IMM}-\text{Degree}}{\text{Degree}}$

(d) VK (e) BA (f) $\Delta_{rel} = \frac{\text{NetShield}-\text{DAVA}}{\text{NetShield}}$

Figure 1015: Results on IM vs. seeds s (a-c); NI vs. blocked nodes k (d-f); Real (RP), Exponential (EP) and Trivalent (TP) probabilities

and False Positive Rates for all values of θ, $\text{TPR} = \frac{TP}{TP+FN}$, $\text{FPR} = \frac{FP}{FP+TN}$. We evaluate the quality of the trained probability model by the Area Under Curve: $\text{AUC} = \int \text{TPR}\, d\text{FPR}$. Figure 1013f presents AUC values for different *content similarity thresholds* τ. We select $\tau = 0.994$, which maximizes AUC (0.9062), as the default τ. Filtering zero-probability edges, under the chosen τ, yields a network of 2094 non-isolated nodes, which we use henceforward.

10.7 Applications

Here, we investigate the behaviour of real-world VK data in comparison to synthetic data generated by the *Barabási-Albert* (BA) model (Figure 1012a) that simulates a power-law degree distribution, on two problems: Influence Maximization (IM), where the objective is to maximize expected spread of selected seeds, and Node Immunization (NI), where the objective is to select nodes to block/remove so that the expected spread of some seeds is minimized. We consider the data-aware NI problem, where seeds are known in advance [291]. We use the algorithm of Holme and Kim [120] that extends the original BA model, selecting parameters so that the BA graph has the same number of nodes and edges, and similar degree distribution (Figure 1013a) and clustering coefficient (0.101) as the VK graph. We evaluate expected spread by 10,000 Monte-Carlo simulations. We also use a synthetic probability model, the trivalency model that randomly select low (0.01), medium (0.05) or high value (0.1), applied to both VK and BA data, and an exponential model on BA data, generating probabilities by the probability density

function $\frac{1}{c}exp(-\frac{x-\mu}{c})$, where c and μ are parameters fit to probability distribution of the VK data. The code and the resulting VK dataset are available[8] online.

Influence Maximization

We apply the IMM algorithm [246] with accuracy parameter $\epsilon = 0.1$ and the Degree heuristic [138] on the IM problem. Figures 1015a-c show our results on different networks and probability models. Δ_{rel} shows the relative performance gap among two algorithms. Revealingly, on VK data, the performance of IMM deviates from that of the naive degree heuristic more with real probabilities than with trivalency (Fig. 1015a), while on synthetic graphs, it deviates more with trivalency (Fig. 1015b, 1015c).

Node Immunization

Node Immunization can be solved in a data-aware manner, when seeds are known in advance, or preemptively, when they are not. We use the state-of-the solution for each case: DAVA [291] accepts a seed set S (calculated by IMM) as input and builds an NI solution informed by *domination* relationships among nodes with respect to S; it calculates the *benefit* of removing a node as $\gamma(v) = 1 + \sum_{u \in \text{children of } v} \gamma(u) \cdot p_{vu}$, where p_{vu} is the probability that influence propagates along the *most probable* path from v to u. NetShield [51] blocks preemptively a set of nodes, S, maximizing the *Shield value*:

$$Sv(S) = \sum_{i \in S} 2\lambda \mathbf{u}(i)^2 - \sum_{i,j \in S} \mathbf{A}(i,j)\mathbf{u}(i)\mathbf{u}(j)$$

where λ and $\mathbf{u}$ are the largest eigenvalue and the corresponding eigenvector of the network's adjacency matrix $\mathbf{A}$. A set S has high Sv if its elements have high eigenscore $\mathbf{u}(i)$ and are not connected to each other (zero $\mathbf{A}(i,j)$). A high eigenscore implies that their removal leads to a significant eigen-drop $\Delta\lambda$. Figures 1015d-f present our results with these algorithms, with $|S| = 100$. Remarkably, DAVA outperforms NetShield in limiting spread all cases except with trivalency on VK. As in IM, using trivalency we overestimate the lead of the state-of-the-art algorithm on BA, and underestimate it on VK.

10.8 Conclusions

In the first part of the chapter, we conducted an exhaustive experimental study of network immunization methods. We conclude that, while data-aware approaches stand out on networks with uniform topologies, spectral structure-based approaches are competitive on networks with power-law topologies. This result calls for further research.

[8] `https://github.com/iconvk/LearningIndependentCascadeOnVK`

In the second part, we presented a framework for extracting influence probabilities for the independent cascade model, using textual content analysis and a vector representation of messages in a network. We showed that our trained model has good prediction power, and applied it experimentally on two network diffusion problems, influence maximization (IM) and network immunization (NI), with real-world and synthetic networks. Juxtaposing results obtained with probabilities derived by our framework to those obtained by synthetic ones, we find that state-of-the art algorithms for IM and NI express their lead with real-world probabilities on real-world networks; the use of a synthetic probability model amplifies that lead on synthetic networks but distorts it on real networks, especially in node immunization.

Chapter 11

On the Robustness of Cascade Diffusion under Node Attacks

How can we assess the ability of a network defined in probabilistic terms to maintain its functionality under failures? *Network robustness* has been studied extensively in the case of deterministic networks under threats to their connectivity. However, applications such as the online diffusion of information and the behavior of networked public raise the question about robustness in a *probabilistic* network. In this chapter, we propose three novel robustness measures for networks hosting a stochastic diffusion process under the Independent Cascade (IC) model, which is susceptible to node failures. The outcome of such a process depends on the selection of its initiators, or seeds, by the *seeder*, as well as on two parameters not on seeder's discretion: the attack strategy and the probabilistic diffusion outcome. In an abstraction, we consider three levels of seeder awareness regarding these two *uncontrolled* parameters, and evaluate the network's viability aggregated over all possible extents of node failures. We introduce novel algorithms from building blocks found in previous works to evaluate the proposed measures. A thorough experimental study with synthetic and real, scale-free and homogeneous networks establishes that the proposed algorithms are effective and efficient, while the proposed measures highlight differences among networks in terms of their robustness and the surprise they can furnish under attack. Last, we devise a new measure of diffusion entropy that can inform the design of probabilistically robust networks.

The content of this chapter was published at the Proceedings of the Web Conference 2020 [174], in co-authorship with Yuchen Li and Panagiotis Karras. The chapter also contains extended results and references, marked by an indentation with a colored bar. Section 11.2 contains an extended literature review on a difference between the Influence Maximization and the Node Immunization problems, and a summary of the submodularity of their objectives under different diffusion models. Section 11.4 contains a new repeatability

experiments of the results of He and Kempe [114]. Section 11.4 presents similarities between proposed measures using Jensen-Shannon divergence. Section 11.4 shows that the difference between two of the measures is the largest for middle-range influence probabilities in a real-world citation network. In Section 11.4 we improve the robustness of Ego-networks in a social network with real-world influence probabilities. We apply our approach of incremental node removal, and illustrate the expressiveness of the newly proposed robustness measure. Finally, Section 11.4 provides an extended study on the scalability of the proposed algorithms. The extended version of the chapter was submitted to IEEE Transactions on Knowledge and Data Engineering [175].

11.1 Introduction

Networks are ubiquitous in the modelling of infrastructures [199], social interactions [165], physical and life-science phenomena [43, 230]. Yet such networks are subjects to *failures* [82], whereby some of their elements may be disabled or removed. The impact of such a structural failure on network performance depends on the desirable features of a network's operation in a particular application.

Network robustness is the ability of a network to retain critical features of its functionality in the face of uncertainty regarding its components. A quantitative measure of robustness expresses the degree in which a network topology may retain essential features despite a failure [229].

Deterministic robustness. Some measures of network robustness gauge the change of a *deterministic* network graph property after *random* failures. The measured property may be the network's diameter, average path length [168], or inverse shortest path length [205]. A critical measure is the size of the largest connected component (LCC) [168, 229], uses in domains from power grids [230] to biological systems [43]. Such analysis is grounded on *percolation theory* [240], which studies problems such as the dependence of a network's largest cluster on node failure probability and predicts phase transitions, i.e., rapid and cardinal changes of network affordance when a parameter reaches a critical value. Schneider et al. [230] proposed a measure aggregating deterministic robustness over all possible sets of blocked nodes to capture a network's vulnerability to failures. Yet random failures have a small effect on *scale-free networks* [7]: an alteration in the periphery does not influence the overall performance much, while a *targeted* attack on hub nodes may easily disconnect the network [230].

Stochastic robustness. In applications such as information diffusion and epidemiology there is uncertainty regarding the *connections* in the network, i.e., the network is *stochastic*. In this context, we may study the operation of such a stochastic network under *targeted* node failures (or, equivalently, attacks on nodes), expressed as the expected number of activated (or infected)

nodes under some parameters of a diffusion process. We refer to this type of robustness as *probabilistic* network robustness. Despite the extensive study of deterministic network robustness [143], its probabilistic counterpart has been only scantily studied. There are studies on how to engineer some kind of robust diffusion in an uncertain or adversarial environment [56, 114], but an investigation on how robustness is to be measured in such environments is missing.

In this chapter, we study the robustness of probabilistic networks expressed by means of the capacity to initiate, in expectation, a successful independent cascade diffusion under adversarial node failures. We introduce three robustness measures, built on two sources of uncertainty: node failures in the network and probabilistic network outcomes.

In more detail, our main contributions are the following:

1. we define the concept of *probabilistic network robustness under adversarial node failures*, which represents the capability of a network to host a diffusion process starting from some seeds, under the Independent Cascade model;

2. we introduce the notion of *seeder awareness* and propose algorithms to measure network robustness under different awareness levels and an unknown number of node failures;

3. we enhance the DAGGER [281] *reachability index* and use it in the case the seeder is aware of network outcomes;

4. we utilize and enhance the runtime of recent solutions to the *Robust Influence Maximization* problem [114] so as to compute the effects of node failures;

5. we study the robustness of scale-free and homogeneous, synthetic and real-world networks, and investigate their interrelationships and values under different parameters.

11.2 Background

Processes in large networks, such as the flow of electricity from supply to customers in a power grid network, package routing and delivery in the internet, and protein delivery in a cell, are vulnerable to failures or attacks. Such events may have dramatic effects, e.g., depriving millions of people from electricity [68], or causing epidemics [261]. There is a need to gauge the extent of such effects and design protection mechanisms. *Network robustness* is defined by the impact of a network perturbation on such processes.

Deterministic Robustness of Integrity

Network robustness reflects a network's ability to maintain its connectivity after the disconnection or deletion of some nodes or edges, targeted (i.e., attacks) or random (i.e., failures) [177]. The connectivity of an *undirected* network is typically measured by the expected size of its largest connected component (LCC) after an attack [177]; maximizing this size renders the network robust. On a deterministic network, even under a probabilistic attack, we obtain a notion of *deterministic robustness*; yet the expected LCC size is also defined on probabilistic undirected networks [123].

The study of robustness under *random failures* is informed by results of *percolation theory*, which describes the physics of *phase transitions* [20, 240] in systems such as those of magnets, fluids [240], and proteins [269], modelled as random networks. A phase transition occurs when certain network parameters pass a *critical threshold* value, causing macro-characteristics to change rapidly [20]. This transition may involve the appearance or destruction of a giant component, as with the spontaneous magnetization of the *Ising magnet* when neighbouring particles have the same magnetic spin under an external field [240, 262]; to determine such transitions, percolation theory examines the expected maximum size [20] of a cluster made of particles in the same *active* state. An common example of the critical threshold is the Molloy-Reed criterion [20],according to which a giant component appears in a general graph if $\frac{\langle k^2 \rangle}{\langle k \rangle} > 2$, where $\langle \cdot \rangle$ denotes averaging and k denotes a node's degree.

The Molloy-Reed criterion shows that scale-free networks are extremely robust to *random node* failures, while being vulnerable to *targeted* node attacks [7]; further, increasing their robustness against targeted attacks is in conflict with maintaining their natural robustness against random failures [20]. Therefore, some robustness measures try to take into consideration both random and target failures, yet do not provide a method to achieve high robustness in those terms [206]. Schneider et al. [230] proposed a local-search heuristic that rewires edges so as to increase an *inclusive* measure of robustness against targeted attacks, while maintaining node degrees, thereby preserving the network's scale-free property (i.e., degree distribution). The said robustness measure is the sum of *worst-case* LCC sizes over all cardinalities of sets of blocked nodes:

$$R(G) = \frac{1}{n^2} \sum_{Q=1}^{n} s(Q) \tag{11.1}$$

where n is the number of nodes in the network and $s(Q)$ is the size of the LCC after removing Q nodes; the normalization by n^2 ensures values are comparable across networks, being in the range $[\frac{1}{n}, \frac{n-1}{2n}]$. This inclusive measure considers all cases of a malicious attack or failure, including those in which the network does not collapse but suffers a big damage [230]. The simple algorithm in [230] leads to an onion-like graph structure, with nodes

of similar degree tending to be connected to each other. Several solutions have been proposed to achieve this property [110]. An LCC-based measure of robustness under *random edge* failures is the *reliability polynomial* [139]:

$$\mathrm{Rel}(G) = \sum_{i=1}^{m} F_i (1-p)^i p^{m-i}$$

where m is the number of edges, F_i the number of sets of i edges whose removal leaves G connected, and p an independent probability that any each edge is present. The closely related problem of securing connectivity between two predefined node sets under edge failures is known as *network reliability* problem [89].

We are interested in the robustness of stochastic diffusion processes under node removals; this notion of robustness resembles the robustness of deterministic networks under targeted node attacks and random edge failures, which has received limited attention [50]. In the next section, we show this connection.

Stochastic Robustness of Diffusion

Apart from the capacity to retain connectivity despite failures, the notion of network robustness also refers to a network's capacity to host a diffusion process, despite the exclusion or *immunization* of some network elements against it, as one may try to contain the diffusion [20, 33, 51, 114]. The mathematical modelling of diffusion is independent of its semantics, i.e., whether that is one of *information*, one of *cascading failures*, or a viral *infection epidemic* [68]; in all cases, the success of the diffusion depends on the network's robustness. Similarly, a node's immunization corresponds mathematically to a node failure, even though the semantics are different. However, now, the effect of node removals is evaluated by a stochastic process, hence the concept of *stochastic robustness* emerges.

Previous work has proposed *epidemic*, *threshold*, and *cascading* diffusion models [287]. There are two popular epidemic models [222]: By the **SIS** model, nodes can be either ***s**usceptible* or ***i**nfected*; a node gets infected from its neighbors by some *infection rate*, stays in that state for some *duration*, and then becomes susceptible again. By the **SIR** model, a node ***r**ecovers* after an infection and becomes immune. The *expected* size of an SIR epidemic starting at u is equal to the expected size of the connected component that contains u [79]. Studies with epidemic models typically consider the infection rate to be homogeneous over a network, yet with information diffusion such rates are heterogeneous [166]. Two models study information diffusion with heterogeneous rates: the *Independent Cascade* (IC) model, a special case of SIR in which each infected node has only one chance to infect others [291], and the *Linear Threshold* (LT) model [138, 165], a stochastic model in which threshold values are uncertain. Under those models, the *Influence Maximiza-*

tion (IM) problem [138] is to select a set of initially active nodes, or *seeds*, so as to maximize the expected total number of activated nodes.

Under the LT-class models, we can solve the IM problem by *minimizing* the expected number of inactive nodes; under the LT model, this turns out to be equivalent to minimizing the expected number of *activated* nodes in the case of a diffusion from a *single* seed [190], i.e., the objective posed by the *Node Immunization* problem in the case of such a diffusion [190]. Under the LT model, selecting seed nodes to maximize diffusion is equivalent to selecting nodes to disable so as to minimize the diffusion from a single seed. That is so because, by LT, only one in-neighbor of a node u can exist in any deterministic instance of a probabilistic graph [138], hence there can be *at most one* directed path from any node v to u. Thus, the expected diffusion $\sigma(v)$ from a single seed v is equal to the sum of probabilities of all paths from v to other nodes. Selecting nodes to block so as to minimize $\sigma(v)$ implies *maximizing* the sum of probabilities of all paths from v that contain selected nodes. If v were a node connected to all others, then the same objective amounts to selecting seed nodes so as to maximize their influence. This relationship implies a connection to the state of an Ising magnet where all particles have the same spin in percolation theory [190], and illustrates the close connection between deterministic and stochastic robustness.

The immunization problem is a special case of the influence-blocking maximization problem (IBM) [55] under the competitive separated-threshold model (STM). In the STM model, nodes pick two independent thresholds for each of two competing opinions in a network. Propagation starts with two sets of seeds, one set per opinion. Let us denote a seed set for a positive opinion as S^+, and for negative as S^-. A node gets activated by an opinion, if a sum of in-neighbours activated by the same opinion exceed the threshold. If a node has enough activated nodes of both opinions, it selects one of the opinions at random (a tie breaking rule). Let σ^+ denote the expected number of nodes activated by the positive opinion. The objective of the competitive IM under the STM model is given by Eq. 11.2, and of the IBM problem by Eq. 11.3, where $\rho^-(S^+, S^-) = \sigma^-(\emptyset, S^-) - \sigma^-(S^+, S^-)$ is a negative influence reduction, that shows how much the existence of positive seeds reduce the negative spread.

$$f_1 = \underset{S^+ \subseteq V \setminus S^-, |S^+|=k}{\arg\max} \sigma^+(S^+, S^-) \tag{11.2}$$

$$f_2 = \underset{S^+ \subseteq V \setminus S^-, |S^+|=k}{\arg\max} \rho^-(S^+, S^-) \tag{11.3}$$

The submodularity of objectives that use the STM model is summarized in Table 111. The IM objective is submodular for both models. The competitive influence maximization (CIM) objective is not submodular for

	IC	LT
IM	Yes [138]	Yes [138]
CIM	No [55]	No [37]
IBM	No for general case [55]	Yes [115]

Table 111: Submodularity of objectives related to the separated-threshold competitive model. Columns correspond to diffusion models, rows to problem objectives. Cells indicate whether an objective is submodular under corresponding model.

the LT and IC modes. The IBM objective is submodular for the LT model, and is not submodular for the general case of the IC model, where edge probabilities are heterogeneous.

Robustness under the IC model

We focus on the IC model, widely used to study word-of-mouth effects in social networks [165], by which a diffusion proceeds in discrete time steps. At time $t = 0$, a set of *seed* nodes $S \in V$ are activated. Any node v activated at time t tries to activate its out-neighbours at time $t + 1$ and succeeds with an independent probability $p_e = p_{uv}$ for each neighbor u. In case of success, the edge e active. This cascading process terminates when there are no more trials for activation. After termination, the set of active nodes and edges, i.e., the single outcome of an IC diffusion, forms a deterministic *live-edge* graph g [138]. The expected cumulative number of activated nodes equals the expected number of nodes reachable from S in G, where each edge may independently fail with probability $1-p_e$. Therefore, the stochastic robustness of diffusion under the IC model corresponds to the deterministic robustness of a *directed* network under targeted node attacks and random edge failures, measured with respect to seeds.

A related question is the IC model's sensitivity to *edge perturbations* [3, 113, 254], whereby, instead of a value of influence probability p_e per edge, there is a confidence interval $[l_e, u_e]$, such that $p_e \in [l_e, u_e]$. Beyond stability, two works study the problem of *robust influence maximization* (RIM) under edge perturbation [56] or any adversarial source of uncertainty [114]. Given a finite set of adversarial strategies $\Theta = \{\theta\}$, the objective in [114] is:

$$\max_{S,|S|\leq k} \min_{\theta\in\Theta} \frac{\sigma_\theta(S)}{\sigma_\theta(S^*_\theta)} \tag{11.4}$$

where $\sigma_\theta(S)$ is the spread (i.e., number of activated nodes) achieved by seed set S under strategy θ and S^*_θ is the optimal seed set for θ, and k is a budget constraint; the normalization by $\sigma_\theta(S^*_\theta)$ emphasizes the fraction of optimal influence achieve, as opposed to an absolute measure; subsequent work [135]

proposed a solution for continuous θ, using the spread function without normalization.

As this objective is not submodular, solutions to the IM problem are not applicable. The Saturate Greedy (SatGreedy) algorithm [114] solves the RIM problem by targeting the cumulative effect of all strategies. This trick makes the objective submodular, since the optimization against a single strategy is submodular. SatGreedy performs a binary search on this objective function. At each step of the search, it constraints the objective value and checks whether it is possible to greedily collect a seed set S so as to reach that objective value constraint. This algorithm is applicable on any monotonic and submodular parameterization of the spread function, and provides a bicriteria approximation guarantee, as violating the budget constraint k by an $O(k \ln |\Theta|)$ factor leads to an $(1-\frac{1}{e})$ approximation of the optimal solution. We adopt the RIM objective as a component in one of the measures we introduce, and apply one of the algorithms in [114] to compute that measure.

Stochastic graph with nodes V and edges E	$G = (V, E)$
Number of nodes and edges of G	n, m
A deterministic graph sampled from G	$g \sim G$
Edge probability parameter	W
A set of immunization strategies	$\Theta = \{\theta_i\}$
Degree of a node v	$d(v \in V)$
Number of blocked (removed) nodes	ℓ
Reachability indicator function	$I(v, S)$
ℓ-sampling parameter	α
EMR-RNI	D
Expected number of activated nodes	σ
A seed set S and size of the set k	S, $k = \|S\|$

Table 112: Notations

11.3 Diffusion Robustness Measures

We propose three novel measures of IC-diffusion robustness, anchored on the abstract awareness of a *seeder*, who selects seed nodes, regarding node failures and probabilistic diffusion outcomes. Table 112 lists our notations.

Immunization Strategies

We measure robustness against node failures, assuming an attacker who disables or immunizes nodes. A consideration of all possible attack strategies amounts to the NP-hard problem of *node immunization* [51, 115, 291], which is outside the scope of this work. We restrict node failures to a strategic set

of structure-aware immunization strategies, $\Theta(\ell) = \{\theta_i(\ell, G)\}$, where ℓ is the number of immunized (disabled, failed) nodes, and G is a directed stochastic network [114]; each $\theta(\ell, G)$ (or $\theta(\ell)$ for brevity) is the set of ℓ nodes chosen by immunization strategy θ applied on G; we denote as g_θ the graph we obtain by removing nodes from a deterministic instance g of graph G according to strategy $\theta(\ell)$. Finally, $\Theta = \{\Theta(\ell)|\ell = \{0..|V| - 1\}\}$. In particular, we opt for attack strategies that select ℓ nodes in an order; each strategy is also a node ranking function. A study on immunization strategies [17] found that a collection of strategies of four algorithmic types are assigned to three or four clusters based on their output by diverse distance measures. We select six strategies that represent each type and cluster in [17], plus a standard spectral-based immunization baseline (NetShield) [51, 171, 227, 291]:

1. *Degree* picks nodes with the largest degree;
2. *Random* picks seed nodes uniformly at random;
3. *Acquaintance* [65] picks a random node's neighbor;
4. *PageRank* ranks nodes by PageRank values [200];
5. *Katz* centrality [137] equals $x_i = \alpha \sum_j \mathbf{A_{ij}} x_j + \beta$, where $\alpha = 0.1$, $\beta = 1$, and $\mathbf{A}$ the network's adjacency matrix.
6. *Betweenness* centrality is the sum of the fraction of all-pairs shortest paths that pass through a node.
7. *NetShield* [51] greedily selects a set of nodes S, aiming to maximize its *Shield value*:

$$Sv(S) = \sum_{i \in S} 2\lambda \mathbf{u}(i)^2 - \sum_{i,j \in S} \mathbf{A}(i,j)\mathbf{u}(i)\mathbf{u}(j)$$

where λ and $\mathbf{u}$ are the largest eigenvalue and the corresponding eigenvector of the adjacency matrix $\mathbf{A}$ containing edge probabilities; λ indicates the effectiveness of a stochastic spread in the network [51]. The algorithm works on undirected networks; we transform any network to undirected by ignoring directions and removing duplicates.

Awareness-based Robustness Measures

We introduce the abstraction of *seeder awareness* regarding attacks and probabilistic diffusion events; this abstraction allows us to study worst-case outcomes. We do not presume that real-world seeders may possess each level of awareness we postulate. We define three notions of robustness based on such worst-case analysis, aggregating outcomes over all possible sizes of immunization attacks, and also a notion of *diffusion entropy* that indicates how much difference seeder awareness can make.

EMR

First, we presume an *omniscient* seeder who has access to an oracle that predicts the outcome g of a diffusion on G as well as any immunization on g that produces g_θ. While knowing these outcomes, an omniscient seeder still needs to take precautions.

As discussed in Section 11.2, the robustness of a deterministic undirected network G can be expressed in terms of the largest connected component (LCC) [177], i.e., the most extensive connected substructure that indicates the largest possible spread (i.e., best possible outcome) of a diffusion emanating from a node in G. When G is a *directed* network, the LCC-equivalent substructure is either of the largest strongly or weakly connected components [232]. Still, none of these two extreme cases generalizing the LCC concept provides information on the maximum number of nodes a seeder can reach, because, in a directed network the exact positioning of seeds within an LCC-like structure matters. We denote the number of nodes that a seeder can reach in an immunized live-edge instance of a directed network, g_θ, with a diffusion from a seed set S of size k, as $\sum_{v \in g_\theta} I(v, S)$, where $I(v, S)$ is a binary function indicating whether there exists a path from S to node v. Maximizing the sum amounts to finding a maximum forest with at most k roots. Let *Expected Maximum Reach* (EMR) be the expected number of nodes an omniscient seeder reaches in G under the worst $\theta \in \Theta(\ell)$:

$$EMR_G(\ell) = \min_{\theta \in \Theta(\ell)} \mathbb{E}_{g_\theta \sim G} \left[\max_{S:|S| \leq k} \sum_{v \in g_\theta} I(v, S) \right] \tag{11.5}$$

Our first robustness measure aggregates $EMR_G(\ell)$ for all values of ℓ, i.e., all cases of a targeted attack or node failure, normalized by network size; we call it *sum of expected maximum reach* or *SEMR*:

$$SEMR_G = \frac{1}{n} \sum_{\ell=1}^{n} EMR_G(\ell) \tag{11.6}$$

To the best of our knowledge, EMF has not been studied hitherto. We introduce a novel algorithm for SEMR computation in Section 11.3 and study its efficiency in Section 11.4.

RNI

We now consider a seeder lacking knowledge of diffusion outcomes, but still having access to an oracle that predicts which nodes will fail. The *maximum* number of nodes such a seeder can expect to reach in G under immunization strategy θ is the maximum, over all cases of S, of the *expected* size, over all instances $g_\theta \sim G$, of the number of nodes $v \in g_\theta$ to which a path exists from S, i.e., $\max_{S:|S|<k} \mathbb{E}_{g_\theta \sim G}[\sum_{v \in g_\theta} I(v, S)]$. A strategy $\theta \in \Theta(\ell)$ that *minimizes* this quantity yields the worst-case number of nodes a seeder reaches, even while

selecting the best S *in expectation*. We call this outcome *Robust Network Immunization* (RNI):

$$RNI_G(\ell) = \min_{\theta \in \Theta(\ell)} \max_{S:|S|\leq k} \mathbb{E}_{g_\theta \sim G} \left[\sum_{v \in g_\theta} I(v, S) \right] \tag{11.7}$$

Our second robustness measure aggregates $RNI_G(\ell)$ for all values of ℓ, normalized by network size. We call this measure *SRNI*:

$$SRNI_G = \frac{1}{n} \sum_{\ell=1}^{n} RNI_G(\ell) \tag{11.8}$$

The computation of RNI requires solving an influence maximization (IM) problem on a graph with $\theta(\ell)$ nodes removed for each immunization strategy $\theta \in \Theta$ and each value of ℓ. We carry out this operation, again building sampled networks g_θ incrementally, using the dynamic influence maximization (DIM) algorithm [198]. DIM extends the TIM algorithm [246] for the dynamic setting, maintaining random samples while the graph structure changes.

RIM

Last, we consider the case of an *agnostic* seeder who has information neither about diffusion outcomes, nor about node failures. The best such a seeder can do is to try to solve a problem of robust influence maximization [114]. We consider the *worst-case* number of nodes a seeder can expect to reach in a stochastic network G with seed set S; that is the minimum, over all immunization strategies $\theta \in \Theta(\ell)$, of the *expected* size, over all instances $g_\theta \sim G$, of the number of nodes $v \in g_\theta$ to which a path exists from S, i.e., $\min_{\theta \in \Theta(\ell)} \mathbb{E}_{g_\theta \sim G}[\sum_{v \in g_\theta} I(v, S)]$. This quantity expresses the *worst-case* spread outcome from the point of view of the spreader.

For the sake of robustness, the spreader should then opt for a seed set S that *maximizes* this worst-case quantity, yielding the *maximum* number of nodes the spreader can expect to reach in G with a seed S under a *worst-case* $\theta \in \Theta(\ell)$ for that S. We call this outcome *Robust Influence Maximization* (RIM):

$$RIM_G(\ell) = \max_{S:|S|\leq k} \min_{\theta \in \Theta(\ell)} E_{g \sim G} \left[\sum_{v \in g_\theta} I(v, S) \right] \tag{11.9}$$

While this quantity is inspired from the objective in of Equation 11.4 in [114], it is based on node removals rather than edge perturbation, and it is *not* normalized by the optimal spread for a given g_θ, as we are interested in robustness in the absolute sense. Our third robustness measure aggregates $RIM_G(\ell)$ for all blocked node set sizes ℓ, normalized by network size. We call this measure SRIM:

$$SRIM_G = \frac{1}{n} \sum_{\ell=1}^{n} RIM_G(\ell) \tag{11.10}$$

To calculate SRIM we apply a sequence of SatGreedy executions [114] with a modified objective, removing the normalization of spread of S by the optimal spread value under strategy θ, S^*_θ:

$$\max_S \rho\prime(S) = \max_S \min_\theta \sigma_\theta(S)$$

The convergence of SatGreedy requires that $\gamma < 1$, where γ is the maximum difference between the binary search upper and lower bounds and the convergence factor (multiplier) for these bounds. If we define ρ' as the minimum over the spread function $\sigma_\theta(S)$, we undo the normalization of upper and lower bounds, and hence of γ, which may then be larger than 1. To prevent that, we normalize ρ by the size of the unblocked network $|V|$:

$$\max_S \rho'(S) = \max_S \min_\theta \frac{\sigma_\theta(S)}{|V|}$$

Further, we enhance the runtime of SatGreedy using the same *dynamic* approach as for SRNI [198] to estimate spread. We also consider the baselines proposed in [114]: *SingleGreedy* selects k seeds sequentially, at each step choosing a seed that maximizes the objective. *AllGreedy* finds the best seed set for each adversary, and selects the one of these that maximizes the objective.

Overview

Our three measures form a sequence, tuning the seeder's awareness regarding the sampling of g and the application of an immunization strategy θ to g by means of two choices: the order of max and min determines whether the seeder is aware of the immunization strategy; the positioning of expectation $\mathbb{E}$ indicates whether the seeder is aware of the sampling of g. Table 113 depicts the relationships among the three measures with respect to these key properties. We observe that one case in the table is not covered by the hitherto described measures, namely the case of a measure corresponding to an unaware spreader, yet with a strategy chosen *posterior* to the sampling of g. This measure, which we call EMinR (Expected *Minimum* Reach), by analogy to the expected maximum reach, expresses the maximum number of nodes in G that a seeder can expect to reach under such a powerful attack:

$$EMinF_G(\ell) = \max_{S:|S|\leq k} \mathbb{E}_{g\sim G} \left[\min_{\theta\in\Theta(\ell)} \sum_{v\in g_\theta} I(v,S) \right] \tag{11.11}$$

We have presented all measures using a worst-case scenario. Still, such measures apply to average cases too. To that end, we need to use an additional expectation function instead of minimization over strategies, resulting in two possible robustness measures, briefly expressed as follows:

$$\max_S \mathbb{E}_\theta \mathbb{E}_G[\sigma] = \max_S \mathbb{E}_{G,\Theta}[\sigma]$$

$$\mathbb{E}_\Theta \left[\max_S \mathbb{E}_{G_\theta}[\sigma] \right]$$

	strategy-aware spreader	agnostic spreader
sampling first	RNI (Eq. 11.7)	RIM (Eq. 11.9)
seeds first	EMR (Eq. 11.5)	*EMinR* (Eq. 11.11)

Table 113: Relationships between robustness measures.

Computation of SEMR

To compute our novel SEMR measure for a single seed, we need to find the expected maximum tree sizes over a sequence of networks g under each immunization strategy $\theta_g \in \Theta$:

$$\mathbb{E}_g[\{T(g \setminus \theta_g(\ell))\}_{\ell=1}^n]$$

We consider immunization strategies θ such that the set of blocked nodes under strategy θ for $\ell + 1$ is a superset of that for ℓ, i.e., $\theta(\ell) \subset \theta(\ell + 1)$. To obtain a sequence of immunization sets $\theta_g(\ell)$ for different values of ℓ on g, it suffices to sequentially remove nodes from g. Equivalently, since we are interested in all values of ℓ, we sequentially add nodes, in reverse. We compute maximum tree sizes over several random samples g from G, with edges pre-sampled and nodes incrementally added, according to strategy θ^*, starting with no nodes, and average values per ℓ to get the expected tree size $EMR(\ell)$. To compute the maximum tree size efficiently, we build upon the DAGGER algorithm [281], employing a dynamic reachability index that returns nodes reachable from any node and also supports node insertions. Given g, the index maintains a directed acyclic graph (DAG), where each node represents a strongly connected component (SCC) in g, called *graph condensation*. A node's insertion implies the insertion of its pre-sampled incident edges. Assume a new edge $e = (u, v)$ is inserted. Let s and t be the SCCs u and v belong to, respectively. DAGGER checks whether there is a path from t to s, using its reachability index. If there is a path, then the insertion of e merges at least two existing SCCs. To find all SCCs to be merged, DAGGER recur-

sively traverses the path from t to s, while pruning descendants of t that do not have a path to s.

We extend DAGGER with a query that returns a network's maximum tree size. Let $g' = (V', E')$ be the DAG that corresponds to g. For each node $v' \in V'$, we maintain a label $v'.r$ as the set of nodes $u' \in V'$ reachable from v': $v'.r = \{u' \in V' | \exists \, \text{path} \, v' \to u'\} \cup \{v'\}$. After inserting a new node w to g, we obtain the corresponding $w' \in g'$, such that w' represents the SCC w belongs to, calculate $w'.r$ based on the out-neighbours of w' in g', and propagate $w'.r$ to all ascendant nodes of w'. Since a new node w may result in the removal of a SCC, we propagate a set of ids of the removed SCCs to ascendant nodes of w' as well. Finally, we maintain a heap of root nodes, valued by the size of their labels. Once an update from w' reaches a node u' with zero in-degree, we add u' to the heap, or update the heap's value if it already contains that w'.

Algorithm 6 illustrates how we compute the SEMR measure incrementally, by calling SEMR() call, which calls INSERT(w, H). H is a heap organizing the nodes of DAG, ordered by the sum of reachable SCC sizes. When performing a node insertion, we first perform the insertion, as explained in the above, by the DAGGER.INSERT() query, then collect the ids of the new node's SCC (Line 3) and all invalidated SCCs (Line 4). Lines 5-6 calculate the reachability of node w', which corresponds to the new node w in the DAG. Lines 8-11 traverse all nodes reachable from w' in the reverse DAG $(g')^T$ by breadth-first search. We update the reachability label of each reverse reachable DAG node u' according to the set of removed SCC's R, and the set of DAG nodes reachable from w'. Last, if during a traversal we reach a root node (Lines 10-11), we insert or update the corresponding reachability value in the heap H.

The SEMR() function in Algorithm 6 returns SEMR for a single seed. For k seeds, in Line 19 we greedily pick k nodes from the heap H, prioritized by marginal gain in terms of reachable nodes in g. We apply the CELF optimization [158] while collecting top root nodes. The CELF optimization is applicable, as (i) adding a new root to a set of selected roots does not change the set of candidate roots, and (ii) the maximization of reachable nodes is a submodular objective, as a special case of the IM objective.

The performance of SEMR computation depends on set union and subtraction operations (Lines 6 and 9). We implement a variant where all operations on reachability labels are performed by a bitset data structure; this measure reduces the time of set operations, but incurs an overhead in calculating the heap value, as it queries single bits for each root node. More advanced set data structures, such as Binary Decision diagrams [144], may improve efficiency further.

Algorithm 6 SEMR Computation

function INSERT(w, H)
    DAGGER.INSERT(w)
    $w' \leftarrow SCC(w)$ ▷ w' is a node in g', that corresponds to SCC in g and has a label r
    $R \leftarrow$ a set of removed nodes from g'
    for all $v'|(w', v') \in g'$ **do**
        $w'.r \leftarrow w'.r \cup v'.r$
    $Q \leftarrow \{u'|\exists \,\text{path}\, w' \rightsquigarrow u' \,\text{in}\, (g')^T\}$
    for all $u' \in Q$ **do**
        $u'.r \leftarrow u'.r \cup w'.r \setminus R$
        if $\nexists v'|(v', u') \in E'$ **then**
            $H.insert(< u', |\{v \in g|SCC(v) \in u'.r\}| >)$
function SEMR
    for all $\theta \in \Theta$ **do**
        $s_\theta \leftarrow$ empty list
        Initialize DAGGER with empty graph
        $H \leftarrow$ a descending heap of $< key, value >$
        for all $v \in \theta$.reverse() **do**
            INSERT(w, H)
            $v', s \leftarrow H.top()$ ▷ Apply CELF here for $k > 1$
            $s_\theta[\ell] = s$
    $s_{\min} \leftarrow$ empty list
    $s_{\min}[\ell] \leftarrow \min_\theta s_\theta[\ell] \,\forall \ell$
    return $\sum s_{\min}$

Sampling ℓ

For the sake of scaling to larger networks, we evaluate our measures for sampled values of ℓ only. As the measures present a rapid decrease in the beginning of the ℓ range, we set a larger sampling rate for smaller ℓ values, and then increase the sample interval geometrically. To that end, we introduce a parameter α that defines the geometric growth of the sampling rate. We sample in steps of

$$\ell \in \{0\} \cup \left\{ \sum_{i=1}^{j} \alpha^{i-1} \right\}_{j=2}^{\lfloor \log_\alpha (1+n(\alpha-1)) \rfloor}$$

If $\alpha = 1$, we sample the complete set of ℓ values. For $\alpha > 1$, we use cubic splines [73] to fit the sampled values and thereby obtain robustness measures for the complete range of ℓ.

Complexity Analysis

Here we analyze the complexity of the presented methods.

SEMR

An iteration of SEMR computation involves DAG maintenance, reachability label propagation, and greedy root selection. The complexity of an edge insertion that does not create a new SCC is constant; in case a new SCC is created, the worst-case complexity is $O(m')$, where m' is the running number edges in the DAG [281]. Reachability label propagation takes $O(m'^2)$, as it updates labels for all ancestors of a new node in the DAG, and each update requires a set union operation on sets of size at most m'. For greedy root selection, it traverses all roots of the DAG and calculates the total size of all SCCs reachable from each root. As we maintain SCC sizes and a list of roots while building the DAG, single root selection takes $O(m')$. We select k roots, resulting in $O(k \cdot m')$, while the CELF optimization makes it significantly faster. With the bitset data structure, there is an additional step to calculate the number of nodes reachable from roots. Each DAG node maintains the number of corresponding SCCs, and a set of reachable DAG nodes. To get the number of reachable nodes, we traverse all bits, incurring an additional m' factor.

To calculate the minimum over all considered immunization strategies, we evaluate *SEMR* per each strategy independently, hence a $|\Theta|$ factor. Summing up, the calculation of *SEMR* takes $O(|\Theta| \cdot nm'(m' + k))$. Using a sampling factor $\alpha > 1$, we perform greedy selection only for sampled nodes. With $q = \lfloor \log_\alpha(1+n(\alpha-1)) \rfloor$ samples, the complexity becomes $O(|\Theta| \cdot (nm'^2 + qk))$.

SRNI

The *SRNI* computation consists of sequential node insertions on the DIM algorithm. In the worst case, we examine all available sketches per node insertion [198]. The minimum number of required sketches is defined by the bound on total weight of sketches $W = \sum_{g\ G}(|g.V| + |g.E|)$, where g is a single sketch sampled from the probabilistic network G, and $g.V$ and $g.E$ are the numbers of nodes and edges in the sampled sketch. After each insertion, DIM performs sampling until $W \geq \beta(n+m)\log n$, where β is an accuracy parameter. In the worst case, g contains all edges of G, while the number of samples may grow to $\beta \log n$. A single sketch may traverse all edges in G, resulting in $O(\beta m \log n)$ complexity per insertion. As before, immunization strategies are independent, yielding a total of $O(|\Theta| \cdot \beta mn \log n)$. The evaluation of spread does not increase the complexity, hence neither does the α parameter.

SRIM

The computation of *SRIM* relies on sequential runs of a robust influence maximization algorithm [114], and incremental maintaining of sketches by DIM [198]. In the worst case, one run of SingleGreedy selects k seed nodes, evaluating the spread marginal gain for each candidate seed and immunization strategy, with $O(kn|\Theta|)$ complexity; with the runtime of DIM, it becomes $O(|\Theta|n(\beta m + k)\log n)$. SatGreedy repeats a similar seed selection as SingleGreedy at each iteration of binary search with accuracy parameter γ, leading to $O(|\Theta|n(\beta m + k)\log n \log \frac{1}{\gamma})$ time.

Approximation Guarantees

To estimate a maximum forest size for EMR, we select root nodes greedily. This method achieves a $(1 - \frac{1}{e})$ approximation guarantee on the maximum forest size in a single MC iteration. By the Central Limit Theorem, increasing the number of iterations, we obtain a $(1 - \frac{1}{e})$ approximation guarantee with respect to the true maximum forest size. RNI inherits the guarantees of DIM [198]. Given a sufficient number of samples (Section 11.3), the algorithm returns a seed set S such that $\sigma(S) \geq \left(1 - \frac{1}{e} - \epsilon\right)\sigma(S^*)$ with probability at least $1 - \frac{1}{n}$, where σ is expected spread, and S^* is an optimal seed set.

The SatGreedy algorithm returns a seed set S' that approximates the original objective $\rho(S)$ with guarantee

$$\rho(S') \geq \left(1 - \frac{1}{e}\right) \cdot \rho(S^*) - \gamma$$

where k is constraint on the number of seeds and $\gamma \in (0, 1)$ is an approximation parameter. For the guarantee to hold, γ has to be related to β as

$$\beta = 1 + \ln|\Theta| + \ln\frac{1}{\gamma}$$

where Θ is the employed set of strategies. It is worth noting that, even with large $\gamma = 0.9$ and only two strategies, $\beta = 1 + \ln 3|\Sigma|/\gamma \approx 2.89$, i.e. SatGreedy requires to increase the seed set size more than 2 times for its approximation guarantee to hold. The authors set $\gamma = 2 \cdot 10^{-3} \cdot |\Sigma|$ empirically [112], while keeping $\beta \leq 2$, therefore the approximation guarantee does not hold for the experiments presented in the paper; the solution operates as a heuristic.

11.4 Experiments

Here we showcase an exhaustive experimental study on the nature and computation of probabilistic network robustness measures. In particular, we study the computation of EMR and RIM measures, for which we have introduced

novel solutions; the computation of RNI relies on a black-box application of the Dynamic Influence Maximization (DIM) algorithm [198], hence we conduct no special computation study therefore. We investigate the nature of all three measures and study their interrelationships and semantics.

In the following we first describe our setup and datasets, then we perform repeatability check of the results of He and Kempe, and study the performance of the algorithm on our objective. We conclude that one of simple baselines (Single Greedy CELF) is more suitable for calculating SRIM, and we use it in further experiments. Next, in Section 11.4 we discuss our a comparative study of three presented measures on various synthetic and real-world data, illustrating patterns of behaviour and expressiveness of the new SEMR measure in comparison to other two. In Section 11.4, we propose a notion of diffusion entropy as a difference between SEMR and SRNI. In the Section 11.4 we show how the expressiveness of SEMR can be exploited to obtain stochastic networks with more robust structure. In the last three sections we study the efficiency of proposed algorithms.

Setup

We ran experiments on an Intel Xeon CPU @ 3.10GHz with 378G RAM running Ubuntu 18.04. All algorithms are implemented in C++ and compiled with gcc 7.4 with -O3 optimization. We set timeout 10h per one measure computation. Runtime and timeout do not include time for the strategy set Θ computation, as the set is equivalent for all robustness measures. We assign edge probabilities either randomly, or uniformly. For random assignment, we pick a value for each edge uniformly from 0 to W, where W is a parameter. For uniform assignment, we assign a certain W value to each edge. We refer to these two types of assignment as *Random* and *Uniform*.

Datasets

Synthetic Networks. We study *power-law networks*, represented by the **Barabási-Albert** (BA) model, and *homogeneous networks*, represented by the **Gaussian Random Partition** (GRP) [41] and **Watts Strogatz** (WS) models. For **BA**, we use the algorithm of Holme and Kim [120], which extends the original Barabási-Albert model, yet use the BA label as its basis. The algorithm randomly creates μ edges for each node in a graph, and for created edge with a probability p adds an edge to one of its neighbors, thus creating a triangle. **GRP** groups nodes so that group sizes follow a Gaussian distribution with expected size s and variance of size equal to s/v, where v is a shape parameter. It uses a probability value p_{in} for edges across nodes in the same group, and p_{out} otherwise. **WS** models self-organizing small-world systems [267], with two parameters: l indicates how many neighbors each node is joined with in a ring; p is a probability of edge rewiring, inducing disorder.

Network	$\|V\|$ $[\cdot 10^3]$	$\|E\|$ $[\cdot 10^3]$	$d_{max}, \overline{d}$	cl
Blogs	1.2	19.0	467, 31	0.336
Minnesota	2.6	3.3	5, 2	0.024
VK	2.8	40.8	288, 29	0.247
Advogato	6.6	47.3	947, 14	0.211
DBLP	12.6	49.7	710, 8	0.117
Brightkite	56.7	212.9	1134, 8	0.117
Gnutella	62.6	147.9	95, 5	0.007
Stanford	281.9	2312.5	38626, 16	0.597

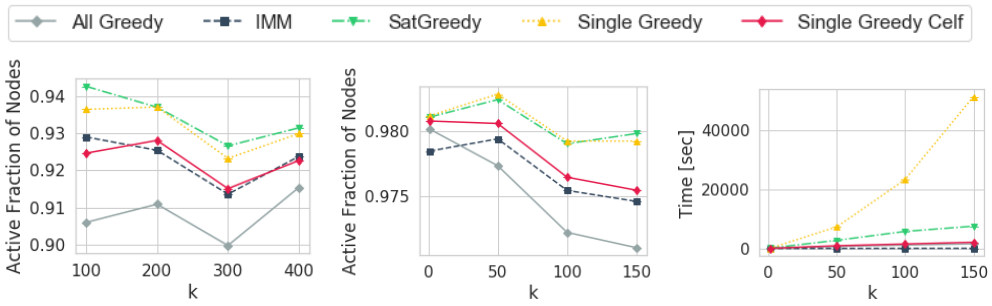
Table 114: Real-world datasets. $d_{max}, \overline{d}$ is maximum and average degree, cl is average clustering coefficient [228].

Real-world networks. We use real-world datasets of various sizes and degree distributions: Blogs contains front-page hyperlinks between blogs during the 2004 US election [2, 151]. DBLP is a citation network of scientific papers [151, 160]. Advogato is a network of trust relationships in an online community platform for free-software developers [151, 185]. Minnesota is a road network [221]. VK is a social network with influence probabilities derived from the content of posts published by users [171]. Brightkite is a location-based social network [64]. Gnutella is snapshots of the Gnutella peer-to-peer file sharing network [157]. Table 114 lists our real-world datasets.

Choice of Algorithm for RIM Computation

As a preliminary experimental choice, we study the performance of methods for RIM calculation, including algorithms and baselines in [114]. We use the IMM algorithm for influence maximization [246] as a non-robust baseline. We include the SingleGreedy method *with* CELF optimization, proposed in [114], and also its variant *without* this optimization, given that, on this non-submodular problem objective, the CELF optimization affects quality.

> We first check the repeatability of the results in [114] on our dataset, including the SingleGreedy method without the CELF speedup. While our metrics are based on Node Immunization, the methods in [114] are tested on edge perturbation, where adversary strategies Θ imply a variation in edge probabilities rather than node removals. We use the WS and GRP networks, with 20 random immunization strategies in Θ. Each strategy forms a network with a complete set of V and E, but randomly reassigns edge probabilities from 0.5 to 1 uniformly. Figure 111 shows our results for growing seed set size k. We confirm that SatGreedy outperforms other methods, as reported in [114]. However, SingleGreedy without CELF achieves as good quality as SatGreedy, albeit with a runtime drawback.



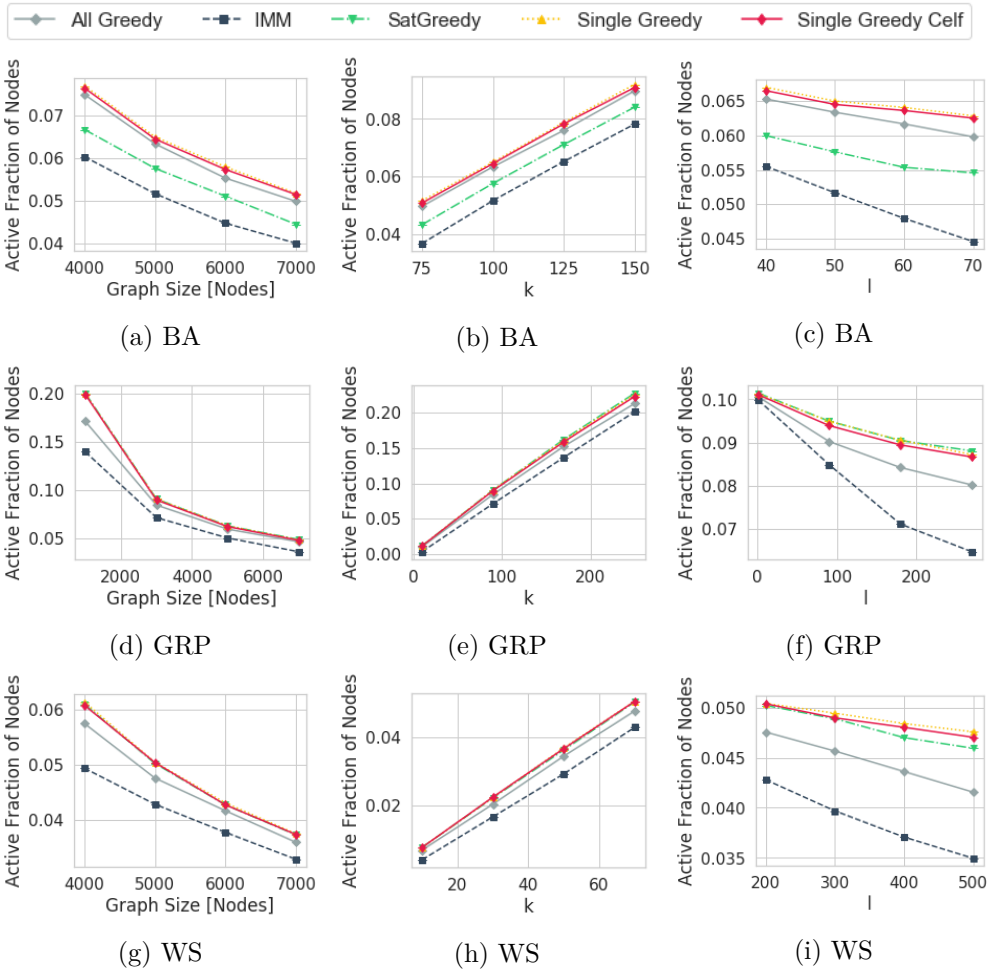
(a) GRP, Objective (b) WS, Objective (c) WS, Runtime

Figure 111: Repeatability of RIM performance under edge perturbation. $n=5000$, $\ell=150$. GRP: $s=100$, $v=0.5$, $p_{in}=0.02$, $p_{out}=5\cdot10^{-4}$. WS: $p=0.3$, $l=15$. SatGreedy accuracy $\gamma=10^{-4}$.

We compare the performance of algorithms under node immunization, i.e., in the computation of the unaggregated RIM objective, with BA, GRP, and WS networks. Figure 112 illustrates the results vs. graph size n, seed set size k, and number of immunized nodes ℓ. The relative performance of algorithms is approximately the same as before, except for that of the non-robust IMM baseline, which does not take uncertainty into consideration and is therefore even more disadvantaged, under node immunization, vs. algorithms that consider adversaries. This disadvantage of IMM grows with ℓ, imprinting the significance of using robust algorithms. SatGreedy is occasionally outperformed by other baselines, especially on BA, while SingleGreedy is consistently leading.

Now we drop the non-robust IMM algorithm out of the comparison, and study the performance of robust algorithms, with the DIM algorithm embedded, on the runtime for computing, and value of, the *aggregate* SRIM robustness measure on the BA network. Figure 113 shows our results for $k=50$ seeds. As in Figure 112, SingleGreedy stands out in terms of objective, at the cost of higher runtime. The difference in objective is more prominent now, as we aggregate the measure over all values from 1 to ℓ. The runtime for computing Θ is negligible, reaching 4s for the largest network.

These results indicate that SingleGreedy (without CELF) offers the best effectiveness, but significantly worse efficiency. SingleGreedy with CELF matches the performance of SingleGreedy under immunization, matches or outperforms that of SatGreedy, is more efficient, and does not require any accuracy parameter γ, as SatGreedy does. Ergo, as we are interested in robustness under node removals, we opt for SingleGreedy *with* CELF in the following.



(a) BA (b) BA (c) BA

(d) GRP (e) GRP (f) GRP

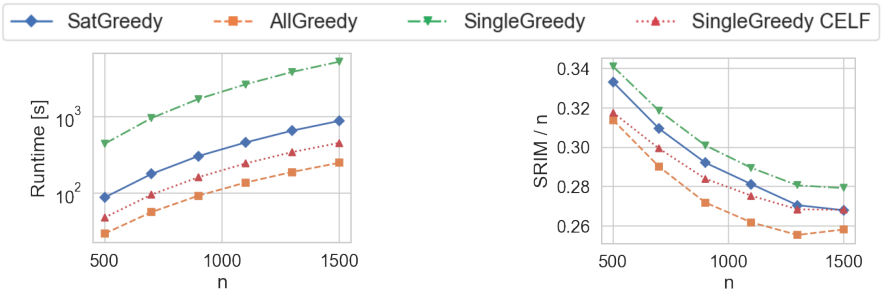
(g) WS (h) WS (i) WS

Figure 112: RIM under node immunization. BA ($n=5\cdot10^3$, $\ell=50$, $k=100$, $\mu=2$), WS ($n=5\cdot10^3$, $\ell=200$, $k=70$), GRP ($n=3\cdot10^3$, $\ell=180$, $k=90$). $W=0.1$, SatGreedy: $\gamma=10^{-4}$.

Measure relationships

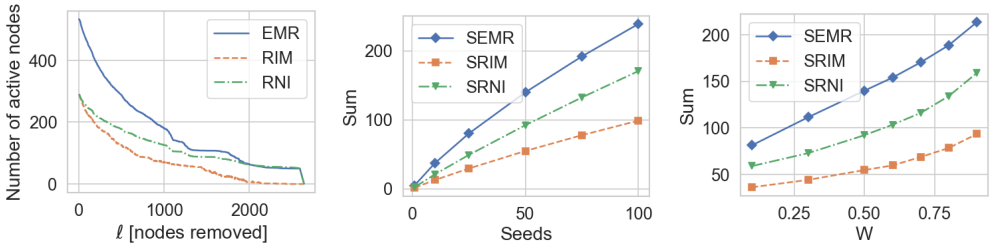
We now study the relation between measures on small networks, and their sensitivity to the set of immunization strategies, using two homogeneous networks (Minnesota and GRP) and two power-law networks (Blogs and VK).

Figure 114a plots plain EMR, RNI, and RIM values, without aggregation, vs. ℓ on Minnesota. Values decrease gradually, revealing some irregularities of graph structure in the middle range of ℓ. EMR and RNI follow a similar pattern, while RIM differs from both. For instance, from $\ell = 1000$ to 2000 EMR and RNI present two abrupt drops at the same value of ℓ. RIM has more, smaller irregularities, but they do not follow EMR and RNI. Figures 114b and 114c present the summed measures (SEMR, SRNI, and SRIM) vs. seed



(a) Runtime of SRIM baselines (b) Objective of SRIM baselines

Figure 113: SRIM on BA, $p=0.4$, $\mu=10$, $W=0.3$, $k=50$.



(a) $k = 50$, $W = 0.8$ (b) $W = 0.5$ (c) $k = 50$

Figure 114: Measures on Minnesota road network.

set size and influence probabilities W, respectively. The difference between measures grows, especially with the size of seed set. Later we will see that a similar trend with seed set size appears in power-law networks (Figures 117c and 119).

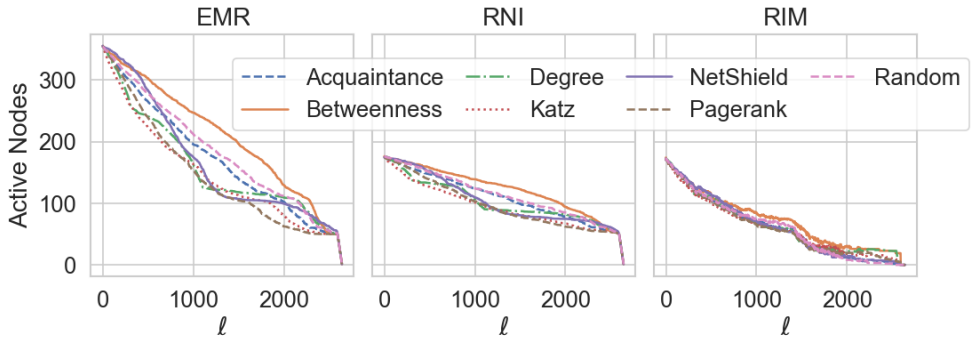
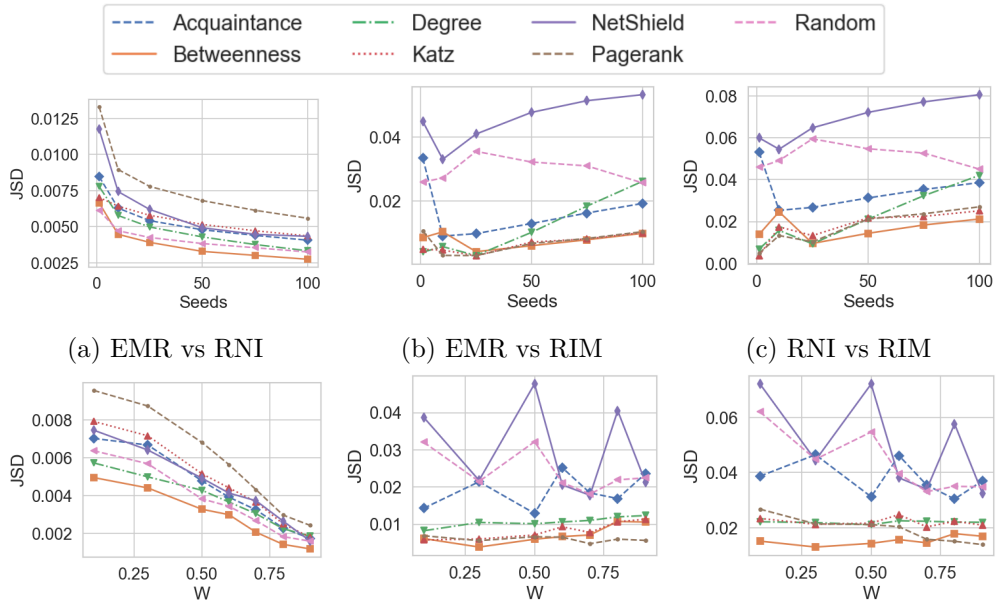


Figure 115: Effect of 7 immunization strategies. Minnesota road network, $W=0.5$, $k=50$, *Random*.

Figure 115 presents a decomposition of measures: instead of taking a minimum over all strategies, we plot the expected influence per strategy, with the

seed set selected by each algorithm. We observe that EMR and RNI follow the same trend *also* for each strategy separately. This is especially conspicuous with NetShield, which shows poor performance in its immunization objective for small values of ℓ, but swiftly improves in the middle range; it then becomes the most effective strategy for a short ℓ range, but looses that position to PageRank. Remarkably, results for RNI presents the same outline, but scaled to a smaller values of active nodes. On the other hand, RIM exhibits a different behaviour, as all strategies mostly produce the same response to the selected seeds. This result illustrates the difference of RIM from the other two measures: RIM is based on the worst case among the complete set of strategies by nature, hence causes the selected seeds to perform almost equally well on any immunization outcome.



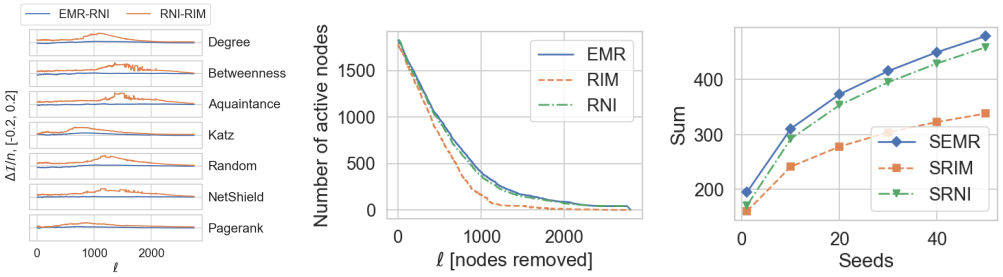
(a) EMR vs RNI (b) EMR vs RIM (c) RNI vs RIM

(d) EMR vs RNI (e) EMR vs RIM (f) RNI vs RIM

Figure 116: JS divergence of measures per strategy.

In Figure 116, we take another view on decomposition of measures: We plot the Jensen-Shannon divergence for each pair of measure distributions over ℓ, and for each strategy, with varying k and W. For example, one point on Figure 116a shows $\mathrm{JSD}(\mathrm{EMR}(\ell) \,||\, \mathrm{RNI}(\ell))$ for a specific k. JSD values for EMR vs RNI are much smaller than for other two pairs, and smoothly converge to zero; values are larger for more effective immunization strategies. On the other hand, the divergence of RIM from both EMR and RNI is unstable and non-monotonic, with diverse trends for different

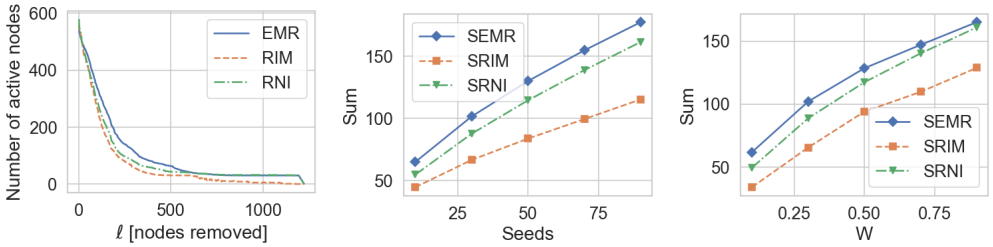
strategies. For example, for Degree and NetShield, JSD grows significantly with number of seeds k, while for Random it drops.



(a) Deltas (b) Measures vs. ℓ (c) Sums vs. k

Figure 117: Dependency of measures on VK social network.

Figure 117a plots the differences EMR-RNI and RNI-RIM vs. ℓ on the VK network. RNI-RIM has a convex shape with a maximum in the middle-range ℓ, while EMR-RNI is almost zero in the whole range. This behavior differs from the one we observed with the BA and DBLP networks, where there is a peak on EMR-RNI. Figure 117b plots non-aggregate measure values for $k = 40$. RNI is very close to EMR along the whole range of ℓ; on the other hand, RNI-RIM also peaks close to the maximum curvature of lines. Figure 117c shows that the effect becomes stronger with larger k, *aggregating* over all ℓ values: SRNI remains close to SEMR, while SRIM diverges from the others; this divergence implies that, on power-law networks, knowledge about immunization, gained when moving from RIM to RNI, is more valuable than knowledge about the stochastic edge outcome, gained when moving from RNI to EMR.



(a) $k=50$, $W=0.8$ (b) $W=0.5$ (c) $k=50$

Figure 118: Dependency of measures, Blogs network. *Random*.

Figures 118 and 119 show the proximity among the three aggregate measures on the Blogs and GRP networks. On the *power-law* Blogs network, the trend is similar to VK, with RNI close to EMR. However, on the *homogeneous* GRP network, RNI is close to RIM for the whole spectrum of network

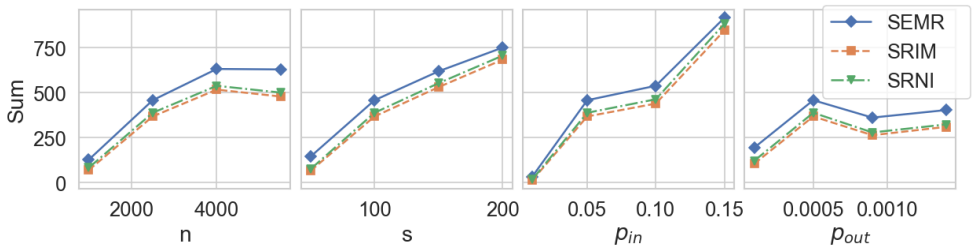
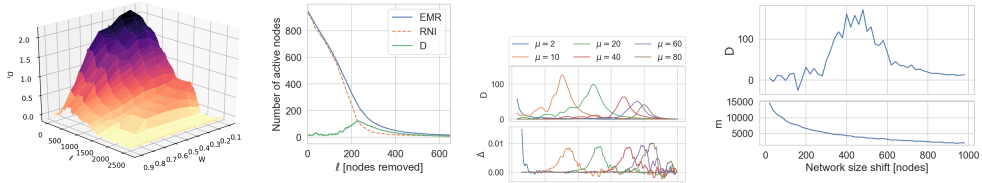


Figure 119: Dependency on network parameters. GRP. $n=2500$, $s=100$, $p_{in}=0.05$, $p_{out}=5\cdot10^{-4}$, $k=10$, $W=0.3$, *Random*.



(a) D_r, Minnesota, $k=10$ (b) Peak Location, BA (c) D vs. Δ, BA (d) Fixed-size network, BA

Figure 1110: Local maximum of D; BA parameters: $n=1000$, $p=0.4$, $\mu=10$; (b): $k=5$; (d): $k=1$, $\Theta=\{\text{Degree}\}$.

shape parameters. We conclude that network topology determines what gain of knowledge matters most; on a homogeneous network, knowledge about a probabilistic outcome is more valuable than knowledge about immunization.

Another interesting feature is the shape of the tail of distribution (Figures 115, 117b and 118a). There exists a value of $\ell=\ell'$, such that all three measures converge to the value of k or ℓ grows towards ℓ', but for $\ell>\ell'$ RIM drops to 0, while others remain at the value of k. Notably, the shape of the drop of RIM is concave, with a gap of first derivative. The region $\ell>\ell'$ corresponds to the case where immunization blocks all nodes by at least one strategy for any seed set. RIM will always select that strategy. However, for EMR and RNI, seeds are selected after the immunization strategy, therefore, for any strategy there are at least k non-blocked nodes.

EMR vs RNI: the diffusion entropy

As we discussed in Section 11.3, the EMR and RNI measures both represent cases in which the immunizer has to prepare for the worst-case, i.e., the case in which the spreader is aware of the immunizer's actions. In other words, both these measures correspond to *robust immunization* problems. Their difference lies in the fact that, under EMR, the spreader is also aware of the probabilistic network outcome. Thus, the difference between these two probabilistic network robustness measures expresses the surprise effect or, so to

speak, *negative entropy* that a probabilistic diffusion outcome can present to the immunizer; it shows how much worse the spread can be in the case of a spreader aware of probabilistic outcomes in comparison to the best guess of a spreader unaware of such outcomes. We study the impact of this difference in more detail, using uniform probability assignment so as to focus on structural effects. We consider the absolute difference D among the two measures; and also the relative difference with respect to RNI, D_r.

$$D = \text{EMR} - \text{RNI}, \quad D_r = \frac{\text{EMR} - \text{RNI}}{\text{RNI}} \tag{11.12}$$

Figure 1110a shows the surface of D_r for different values of ℓ and W on the Minnesota network. D_r is larger for smaller number of removed nodes ℓ, and drops with larger edge probabilities. Still, it is not monotonic vs. W; it obtains a maximum value around $W = 1.5$, and the peak is more explicit with smaller ℓ. Figure 1110b shows that this non-monotonic behavior of D_r also appears with respect to ℓ on a BA network, and indicates exactly where the peak is located. Compared to Figure 1110c, where peaks are presented only for a single seed, we see that on Figure 1110b the peak has larger width.

We have observed that D also relates to the relative marginal gain seeds addition by the spreader. We define $\delta_{\theta_i}(\ell)$ as the relative marginal gain of the second seed for any strategy $\theta_i \in \Theta$ under ℓ immunized nodes:

$$\delta_{\theta_i}(\ell) = \frac{\max_{S:|S|=1} \sigma_{\theta_i(\ell)}(S) - \max_{S:|S|=2} \sigma_{\theta_i(\ell)}(S)}{\max_{S:|S|=1} \sigma_{\theta_i(\ell)}(S)}$$

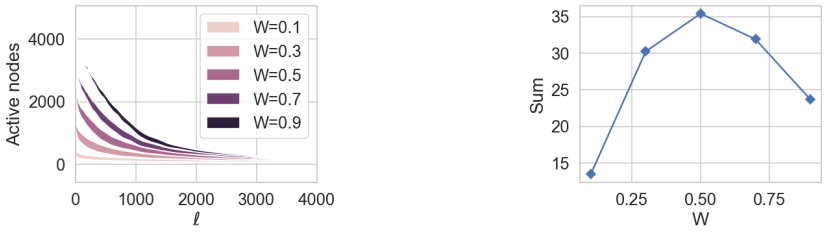
We then calculate a new quantity $\Delta(\ell)$ as the maximum differential quotient of δ over all strategies for each ℓ:

$$\Delta(\ell) = \max_{\theta_i \in \Theta} \{\delta_{\theta_i}(\ell) - \delta_{\theta_i}(\ell - 1)\}$$

Figure 1110c juxtaposes D and Δ, plotted with moving average smoothing. Remarkably, their two peaks align, with a slight shift to the right for Δ. This finding implies that, on BA networks, the values of ℓ for which the network ceases to be strongly centralized, hence Δ flattens out, would also cause the highest surprise to an immunizer.

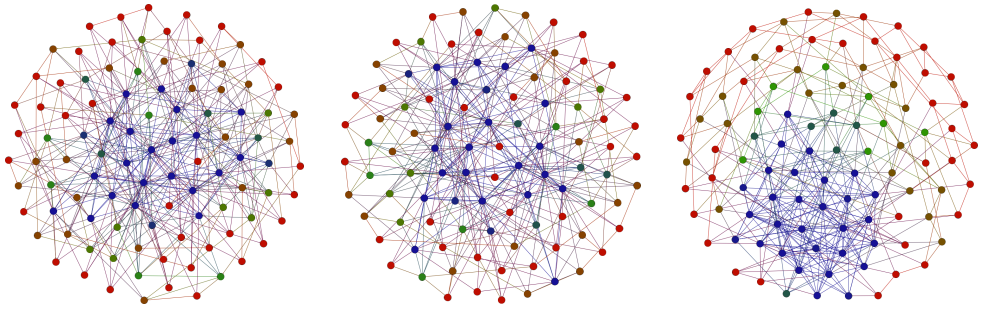
We exploit this observation to generate networks of *enhanced robustness*: we fix size to 1000 nodes, yet first generate a network of larger size and then remove superfluous nodes by the Degree strategy. We call the amount of nodes first added and then removed *shift*. Figure 1110d plots D vs. shift. Shifting improves network robustness in terms of D; we create networks in which a seeder has the potential to perform surprisingly well against an immunizer. The lower subfigure plots the number of edges in the obtained network; as there is no correlation between the peak of D and number of edges, the peak must be attributed to the network's structure.

We also plot D as a colored interval vs. ℓ, on DBLP, while varying edge probabilities. Figure 1111a shows the results. D is the largest (i.e., widest) for middle-range W values. Figure 1111b illustrates this fact for a single ℓ. This non-monotonic dependence of D on weights suggests controlling a network's robustness by tuning edge weights.



(a) EMR vs RNI; color indicates difference. (b) D (SEMR-SRNI)

Figure 1111: The difference between EMR and RNI on DBLP. $\alpha = 1.075, k = 50, \beta = 40$, *Random*.



(a) Original (b) SRIM-based (c) SEMR-based

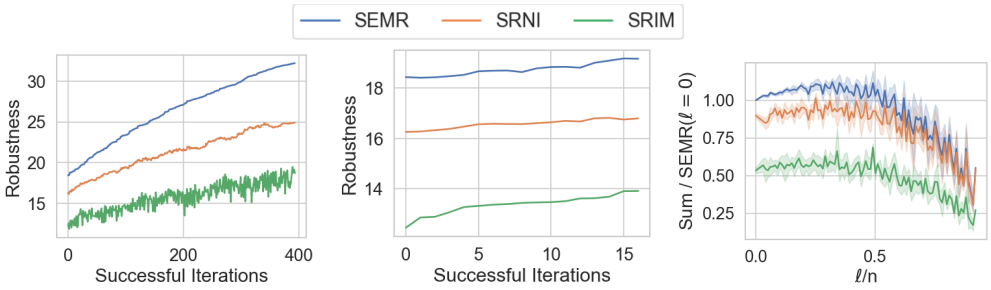
Figure 1112: Robust BA networks

Case Studies of Network Robustness

We provide examples of robust networks using the local search heuristic of [230], which randomly samples pairs of edges, e.g., pair $\{(v_1, u_1), (v_2, u_2)\}$, and rewires them to $\{(v_1, u_2), (v_2, u_1)\}$ if that leads to a higher robustness measure. We experiment with SRIM and SEMR, since SRNI exhibits similar behavior to SEMR (see Section 11.4). The sampling proceeds until $|E|$ iterations bring no change.

We experiment with a random BA network of 100 nodes, uniform edge probability of 0.5, and 2 seeds. Figure 1112 shows the original network (non-robust), and two networks obtained by the aforementioned procedure

for SRIM and SEMR, respectively. Colors indicate similar node degrees, blue for larger, green for medium, and red for smaller. We plot the networks using the Fruchterman Reingold algorithm [25]. We note that the network targeting SEMR has a layered onion-like structure, similar to robust static networks [110], while the other two networks do not show evident patterns.



(a) BA, SEMR-based (b) BA, SRIM-based (c) VK, Ego

Figure 1113: Effect of rewiring (a,b), node removals (c)

Figure 1113 illustrates all robustness values for the three networks throughout the algorithm's successful iterations. Figure 1113a shows the case targeting SEMR; as expected, SEMR grows monotonically, while SRNI is almost monotonic. Contrariwise, SRIM is oscillating, albeit with a rising trend. Due to this oscillation, the algorithm targeting SRIM does not find good changes in the network, as we witness on Figure 1113b, in which there are only 16 successful iterations in total, and the robustness improvement is minor. This result highlights SEMR as a more expressive measure.

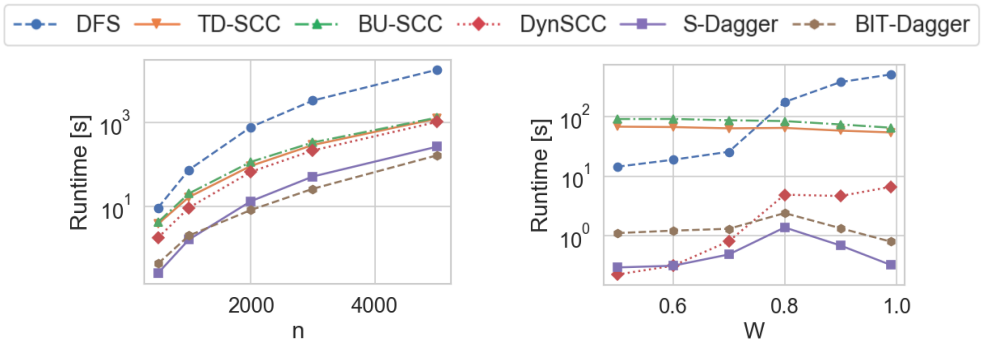
In another approach, rather than rewiring edges, we incrementally remove nodes that lead to the highest marginal gain in SEMR. We apply this approach to ego networks of VK with real-world influence probabilities, as in [171]. The ego network of node v contains v (*ego*), adjacent nodes of v (*alters*), and all edges between them. Ego networks are useful in understanding micro-level structures of social networks [12]. We sample ego networks of size between 10 and 50 nodes, having from 10 to 300 edges. Since ego networks differ in size and shape, we normalize results by the value of SEMR at $\ell = 0$. Figure 1113c presents our results. The solid line is averaging over 100 samples, and the shaded region represents standard deviation; the x-axis is normalized by the total number of nodes in a network. We see that applying the greedy node removal leads to an average $\sim 10\%$ of increase of SEMR after removing $\sim 30\%$ of nodes, and a positive improvement for any number of nodes until $\sim 50\%$ of all nodes removed, while other measures are less affected.

SEMR Computation Efficiency

Here, we study the efficiency in computing SEMR, which is the most novel of the measures we propose. The implementation is based on the DAGGER reachability index [281]. We compare the runtime of S-Dagger, which uses a set data structure, and BIT-Dagger, which uses a bitset data structure, to the following baselines, which progressively introduce algorithm features:

- DFS finds the maximum tree by depth-first search for each node in each MC iteration;
- TD-SCC performs a deterministic graph condensation (i.e., finds SCCs that form a directed acyclic graph (DAG)) and runs a *top-down* breadth-first search from each root node in the DAG to find the maximum tree;
- BU-SCC performs a deterministic graph condensation with *bottom-up* reachability labelling, similarly to S-Dagger, but without a dynamic reachability index;
- DynSCC performs graph condensation with *dynamic* bottom-up labeling, but instead of the Dagger index, it maintains DAGs naively, decreasing ℓ and rerunning Tarjan's algorithm [248] for each affected DAG node.

We study performance on BA with growing size n and Minnesota with growing weigths W, using the Degree immunization strategy and 10 iterations to estimate runtimes. Figure 1114 presents our results. The immunization runtime for $\Theta = \{Degree\}$ is trivial.



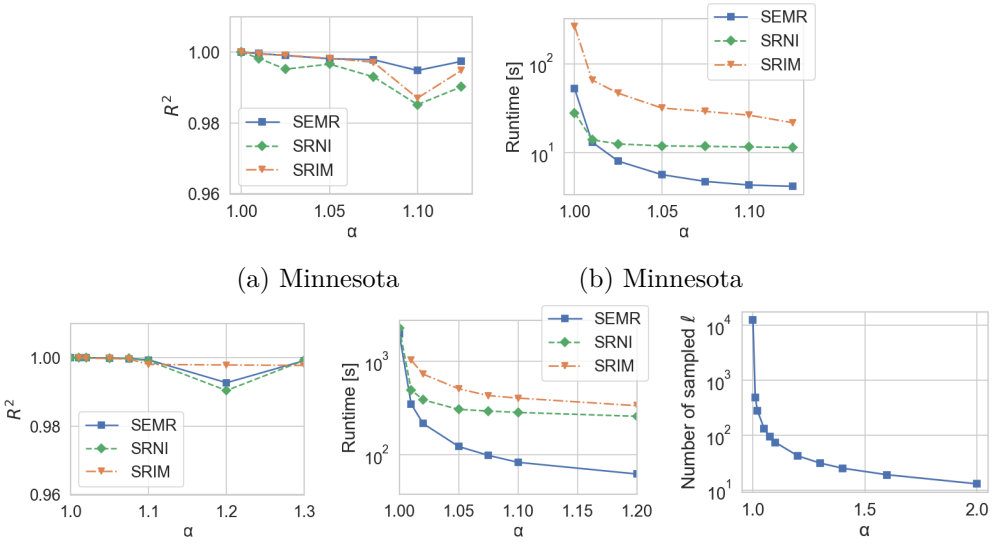
(a) BA, $W = 0.3, p = 0.4, \mu = 10$.

(b) Minnesota road network

Figure 1114: EMR computation. $\Theta = \{\text{Degree}\}$, $k = 1$.

DAGGER algorithms achieve a significant runtime improvement in comparison to baselines. BA (Figure 1114a) is a denser, power-law network, while Minessota (Figure 1114b) is a sparse homogeneous network. In both cases, DFS is the worst approach; graph condensation significantly improves runtime. On Minessota, the runtime of TD-SCC and BU-SCC even improves as

weight W grows, as more SCCs appear. On this sparse network, the efficient maintenance of SCCs is crucial. DynSCC maintains SCCs less efficiently than DAGGER, hence its runtime deteriorates as W grows. BIT-Dagger is less efficient than S-Dagger on the sparse graph, though more efficient on the dense power-law network, as traversing the labels of each root node and retrieving SCC sizes corresponding to reachable nodes is much less costly for sets than bitstrings. On a sparse graph, the bitset data structure incurs a large overhead traversing sparse bit strings. Still, on a dense graph, the bitset structure compensates by significantly more efficient set operations. Henceforward, we use S-Dagger as the default option.



(a) Minnesota (b) Minnesota

(c) DBLP (d) DBLP (e) # of sampled ℓ

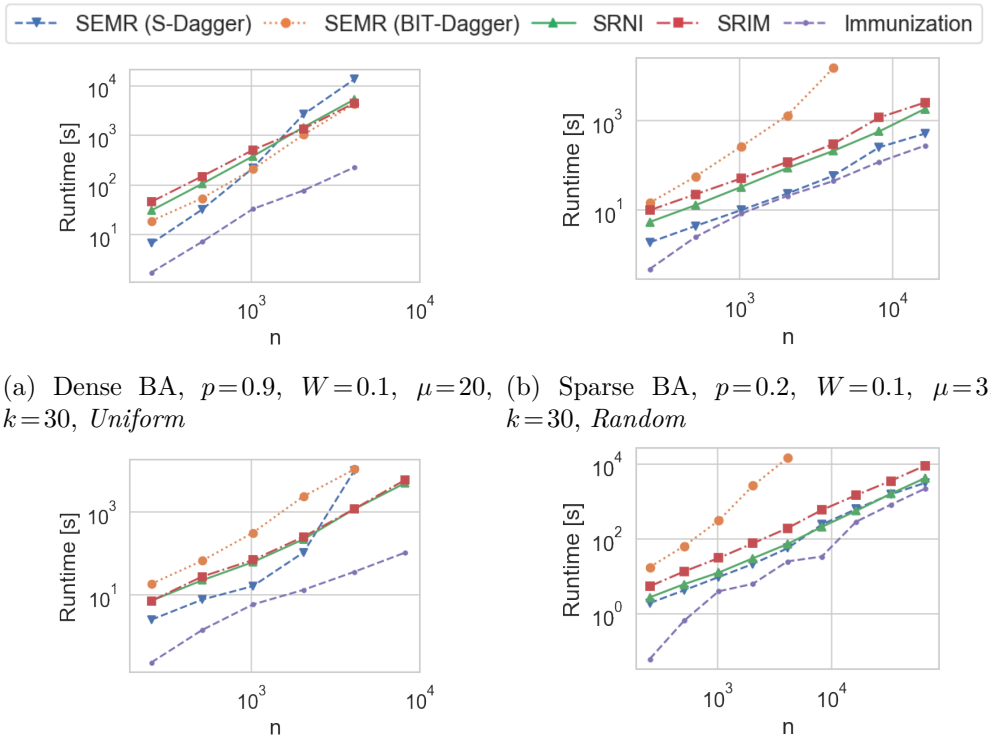
Figure 1115: Performance vs. α; R^2 averaged over adversaries, *Random* prob; Minnesota $W=0.7$, $k=10$; DBLP $W=0.5$, $k=50$; Immunization time 2s on Minnesota, 82s on DBLP.

Sampling Accuracy

We now investigate the accuracy of our methods using the sampling approach of Section 11.3. Figure 1115 shows the effect of the α parameter on performance, on Minnesota and DBLP networks (power-law and homogeneous, respectively). As an accuracy measure, we use the coefficient of determination R^2 between measures calculated for each value of ℓ (observation) and measures calculated using cubic splines over sampled values of ℓ (model). On the DBLP dataset, SRIM does not terminate for $\alpha = 1$, so we use a fitted model for $\alpha = 1.01$ as ground truth. On both networks, α increasing from 1 to 1.075 does not affect the accuracy of fit significantly, even while it allows

the runtime of all algorithms to drop rapidly. Yet accuracy drops for $\alpha = 1.1$ on Minnesota and $\alpha = 1.2$ on DBLP. Given these results, we select $\alpha = 1.075$ as a default value in the following.

Scalability



(a) Dense BA, $p=0.9$, $W=0.1$, $\mu=20$, $k=30$, *Uniform*

(b) Sparse BA, $p=0.2$, $W=0.1$, $\mu=3$, $k=30$, *Random*

(c) Dense GRP, $W=0.6$, $k=50$, $s=60$, $v=0.5$, $p_{in}=0.015$, $p_{out}=5 \cdot 10^{-4}$, *Rand*.

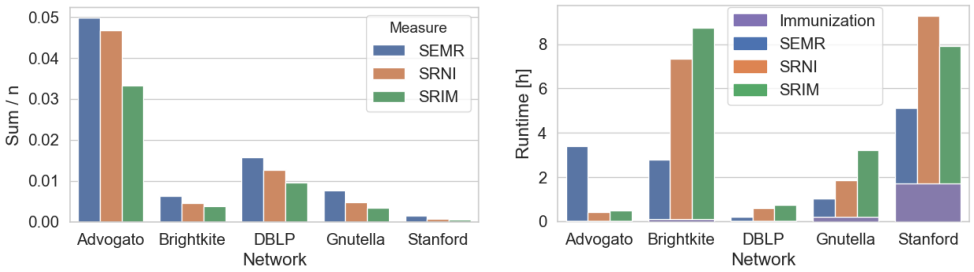
(d) Sparse GRP, $W=0.1$, $k=50$, $s=40$, $v=0.5$, $p_{in}=0.01$, $p_{out}=10^{-4}$, *Random*

Figure 1116: Scalability on synthetic networks. $\alpha = 1.075$.

We investigate the effect of sparsity of diffusion on the performance of algorithms. Figure 1116 shows our results. Notably, in networks where the fraction of active nodes is close to 1, the efficiency of SEMR is very low, and it becomes the slowest out of all three algorithms. At the same time, SRNI and SRIM perform similarly. In order to deal with such high-density networks, we switch to the BIT-Dagger implementation of SEMR, that leads to more scalable runtime growth than S-Dagger (Figure 1116c). The violet line represents the immunization time, common for all measures.

Last, Figure 1117 shows runtime and aggregate measures *normalized* by n for larger networks. Our three measures present a consistent order, while

their runtimes depend on the density and size of networks. Advogato has much higher ratio of active nodes, so SEMR performs poorly. In reverse, in Brightkite the relative number of active nodes is small, so SEMR is faster than other two measures. DBLP has a larger fraction of active nodes than Brightkite and Gnutella, yet its runtime keeps low, due to its more *flat* structure, i.e., smaller edge density and maximum degree. We also compare relative difference of SEMR vs SRNI, and SRNI vs SRIM. Advogato shows a larger increase of spread as the spreader becomes aware of the immunization strategy (i.e., from SRIM to SRNI); on Gnutella the information about particular spread outcome brings slightly more profit. On the other networks the two differences appear very similar. For the Stanford network, SRNI and SRIM did not terminate within 10h for $W = 0.5$ and $\alpha = 1.075$, hence we decreased the problem complexity by setting $W = 0.1$ and $\alpha = 1.2$, therefore the reported fraction of active nodes is low.



(a) Robustness (b) Runtime

Figure 1117: Scalability of larger networks. $W = 0.5$, *Random*, $k = 100$, $\alpha = 1.075$; on Stanford: $W = 0.1$, $\alpha = 1.2$

11.5 Conclusions

We introduced three aggregate measures that evaluate the diffusion robustness of probabilistic networks. We anchor these measures on a spreader who orchestrates an Independent Cascade diffusion under node failure attacks. Each measure is based on a notion of worst-case maximum expected spread. We introduced efficient algorithms to calculate these measures and sample-based versions thereof that enable their computation on realistic networks of up to 10^5 nodes. Our experimental study determined that, on scale-free networks, measures sharing the same notion of seeder awareness regarding the adversarial attack are closer, while those sharing the same notion of awareness regarding the network instance are closer on homogeneous networks. Our results provide tools for assessing the robustness of real-world probabilistic networks, and offer guidelines on how to design robust networks and enhance the robustness of existing ones.

Bibliography

[1] R. Aboolian, O. Berman, and D. Krass. Competitive facility location and design problem. *European Journal of Operational Research*, 182(1):40–62, 2007. 22

[2] L. A. Adamic and N. Glance. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43, 2005. 141

[3] A. Adiga, C. J. Kuhlman, H. S. Mortveit, and A. Vullikanti. Sensitivity of diffusion dynamics to network uncertainty. *Journal of Artificial Intelligence Research*, 51:207–226, 2014. 129

[4] R. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical programming*, 53(1-3):243–266, 1992. 11

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows: Theory. *Algorithms, and Applications*, 526, 1993. 10, 11

[6] R. K. Ahuja, M. Kodialam, A. K. Mishra, and J. B. Orlin. Computational investigations of maximum flow algorithms. *European Journal of Operational Research*, 97(3):509 – 542, 1997. 11

[7] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378, 2000. 124, 126

[8] S. Alumur and B. Y. Kara. A new model for the hazardous waste location-routing problem. *Computers & Operations Research*, 34(5):1406–1423. 22

[9] American Statistical Association. ASA Statement on the Role of Statistics in Data Science. AMSTATNEWS. `https://magazine.amstat.org/blog/2015/10/01/asa-statement-on-the-role-of-statistics-in-data-science/`, 2015. Online; accessed 20 January 2019. 3

[10] E. Anshelevich, D. Chakrabarty, A. Hate, and C. Swamy. Approximability of the firefighter problem: Computing cuts over time. *Algorithmica*, 62(1-2): 520–536. 3

[11] C. Araz, H. Selim, and I. Ozkarahan. A fuzzy multi-objective covering-based vehicle location model for emergency services. *Computers & Operations Research*, 34(3):705–726, 2007. 22

[12] V. Arnaboldi, M. Conti, M. L. Gala, A. Passarella, and F. Pezzoni. Ego network structure in online social networks and its impact on information diffusion. *Computer Communications*, 76:26–41, 2016. 150

[13] L. M. Ausubel, P. Milgrom, et al. The lovely but lonely vickrey auction. *Combinatorial auctions*, 17:22–26, 2006. 28

[14] Y. Aviv and A. Pazgal. A partially observed markov decision process for dynamic pricing. *Management science*, 51(9):1400–1416, 2005. 29

[15] H. Aziz, S. Gaspers, S. Mackenzie, and T. Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015. 81

[16] M. Babaioff, S. Dughmi, R. Kleinberg, and A. Slivkins. Dynamic pricing with limited supply. *ACM Transactions on Economics and Computation (TEAC)*, 3 (1):1–26, 2015. 27, 28

[17] M. B. Baig and L. Akoglu. Correlation of node importance measures: An empirical study through graph robustness. In *WWW Conference Companion*, pages 275–281, 2015. 131

[18] S. R. Balseiro, D. B. Brown, and C. Chen. Dynamic pricing of relocating resources in large networks. *ACM SIGMETRICS Performance Evaluation Review*, 47(1):29–30, 2019. 29

[19] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 31–40, 2006. 81

[20] A.-L. Barabási. *Network science*. Cambridge university press, 2016. 126, 127

[21] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. *Knowledge and Information Systems*, 37(3):555–584. 38, 114

[22] J. K. Barker and R. E. Korf. Limitations of Front-To-End Bidirectional Heuristic Search. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1086–1092, 2015. 24

[23] S. Barman and S. K. K. Murthy. Approximation algorithms for maximin fair division. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 647–664, 2017. 81

[24] S. Barman, G. Ghalme, S. Jain, P. Kulkarni, and S. Narang. Fair division of indivisible goods among strategic agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1811–1813, 2019. 81

[25] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009. 84, 150

[26] B. Behsaz, M. R. Salavatipour, and Z. Svitkina. New approximation algorithms for the unsplittable capacitated facility location problem. *Algorithmica*, 75(1): 53–83, 2016. 22

[27] O. Berman, D. Krass, and Z. Drezner. The gradual covering decay location problem on a network. *European Journal of Operational Research*, 151(3):474–480, 2003. 22

[28] O. Berman, V. Verter, and B. Y. Kara. Designing emergency response networks for hazardous materials transportation. *Computers & operations research*, 34 (5):1374–1388, 2007. 22

[29] D. P. Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21(1):152–171, 1981. 11

[30] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 1, pages 560–564. 1995. 18

[31] S. Bird, E. Klein, and E. Loper. *NLP with Python*. O'Reilly, 2009. 117

[32] G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203–229, 2003. 26

[33] I. Bogunovic. Robust protection of networks against cascading phenomena. Master's thesis, Department of Computer Science, ETH Zürich, 2012. 127

[34] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2017. 117

[35] E. Bokányi and A. Hannák. Ride-share matching algorithms generate income inequality. *CoRR*, abs/1905.12535v1, 2019. 78, 79

[36] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 946–957. 2014. 40

[37] A. Borodin, Y. Filmus, and J. Oren. Threshold models for competitive influence in social networks. In *Internet and Network Economics - 6th International Workshop*, pages 539–550, 2010. 129

[38] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*, pages 421–430, 2001. 3

[39] A. Boskovic, Q. Chen, D. Kufel, and Z. Zhou. Online learning and matching for resource allocation problems. *CoRR*, abs/1911.07409, 2019. 3, 16, 33

[40] M. K. Boujelben, C. Gicquel, M. Minoux, M. Kchaou Boujelben, C. Gicquel, and M. Minoux. A MILP model and heuristic approach for facility location under multiple operational constraints. *Computers & Industrial Engineering*, 98:446–461. 22, 23

[41] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *European Symposium on Algorithms*, pages 568–579, 2003. 108, 140

[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. 90

[43] N. E. Brunk, L. S. Lee, J. A. Glazier, W. Butske, and A. Zlotnick. Molecular jenga: the percolation phase transition (collapse) in virus capsids. *Physical Biology*, 15(5):056005, 2018. 124

[44] C. Budak, D. Agrawal, A. E. Abbadi, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674. 2011. 38

[45] E. Budish. The combinatorial assignment problem: approximate competitive equilibrium from equal incomes. In *Proceedings of the Behavioral and Quantitative Game Theory - Conference on Future Directions*, page 74:1, 2010. 81

[46] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems, revised reprint*, volume 106. Siam, 2012. 10, 11, 12

[47] J. C. Castillo, D. Knoepfle, and G. Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, page 241242, 2017. 28

[48] D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C. Wong. Subquadratic submodular function minimization. In H. Hatami, P. McKenzie, and V. King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1220–1231. 2017. 17

[49] H. A. Chaudhari, J. W. Byers, and E. Terzi. Putting data in the driver's seat: Optimizing earnings for on-demand ride-hailing. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 90–98, 2018. 28, 78, 79, 95

[50] L. Chayes, R. H. Schonmann, et al. Mixed percolation as a bridge between site and bond percolation. *The Annals of Applied Probability*, 10(4):1182–1196, 2000. 127

[51] C. Chen, H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. Node immunization on large graphs: Theory and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):113–126, 2016. 101, 102, 107, 120, 127, 130, 131

[52] F. Chen, H. Lin, Y. Gao, and D. Lu. Capacity constrained maximizing bichromatic reverse nearest neighbor search. *Expert Systems with Applications*, 43: 93–108, 2016. 40, 51

[53] H. Chen, Y. Jiao, Z. T. Qin, X. Tang, H. Li, B. An, H. Zhu, and J. Ye. InBEDE: Integrating contextual bandit with TD learning for joint pricing and dispatch of ride-hailing platforms. In *Proceedings of the 2019 IEEE International Conference on Data Mining*, 2019. 27, 28, 29

[54] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038. 2010. 22

[55] W. Chen, L. V. S. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013. 128, 129

[56] W. Chen, T. Lin, Z. Tan, M. Zhao, and X. Zhou. Robust influence maximization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 795–804, 2016. 125, 129

[57] Z. Chen, Y. Liu, R. C.-W. Wong, J. Xiong, G. Mai, and C. Long. Optimal Location Queries in Road Networks. *ACM Transactions on Database Systems*, 40(3):17:1—-17:41. 3, 22, 23, 49, 51

[58] J. Cheng, L. A. Adamic, P. A. Dow, J. Kleinberg, and J. Leskovec. Can Cascades be Predicted? In *Proceedings of the 23rd International Conference on World Wide Web*, pages 925–936, 2014. 114

[59] S. Cheng and Y. Mao. Restricted max-min fair allocation. In *45th International Colloquium on Automata, Languages, and Programming*, pages 37:1–37:13, 2018. 81

[60] C.-I. Chiang, M.-J. Hwang, and Y.-H. Liu. An alternative formulation for certain fuzzy set-covering problems. *Mathematical and computer modelling*, 42 (3):363–365, 2005. 22

[61] W.-C. Chiang, J. C. Chen, and X. Xu. An overview of research on revenue management: current issues and future research. *International journal of revenue management*, 1(1):97–128, 2007. 26, 27

[62] Chicago Data Portal. Taxi trips. `https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew`, 2019. Online; accessed 6 August 2019. 82, 83

[63] Chicago Data Portal. Census Boundaries. `https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Census-Tracts-2010/5jrd-6zik`, 2019. Online; accessed 6 August 2019. 83, 84

[64] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011. 141

[65] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Physical review letters*, 91(24):247901, 2003. 131

[66] Constantinos Daskalakis. 6.896 Topics in Algorithmic Game Theory. Lecture Notes. `https://people.csail.mit.edu/costis/6896sp10/lec2.pdf`, 2010. Online; accessed 6 August 2019. 41

[67] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. The Uncapacitated Facility Location Problem. *Discrete Location Theory*, pages 119–171, 1990. 50

[68] P. Crucitti, V. Latora, and M. Marchiori. Model for cascading failures in complex networks. *Physical Review E*, 69(4):045104, 2004. 125, 127

[69] W. Cui, X. Gong, C. Liu, D. Xu, X. Chen, D. Fang, S. Tang, F. Wu, and G. Chen. Node Immunization with Time-Sensitive Restrictions. *Sensors*, 16 (12):2141. 40, 101, 103, 110, 113

[70] G. Dai, J. Huang, S. M. Wambura, and H. Sun. A balanced assignment mechanism for online taxi recommendation. In *18th IEEE International Conference on Mobile Data Management*, pages 102–111, 2017. 42, 78, 79

[71] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment*, 10(3): 85–96. 25

[72] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951. 14

[73] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag GmbH, 2001. 137

[74] K. Deng, S. W. Sadiq, X. Zhou, H. Xu, G. P. C. Fung, and Y. Lu. On group nearest group query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):295–308, 2012. 49

[75] U. Derigs. A shortest augmenting path method for solving minimal perfect matching problems. *Networks*, 11(4):379–390, 1981. 9, 10, 52, 55

[76] E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 24

[77] J. Djolonga and A. Krause. Differentiable learning of submodular models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1014–1024, 2017. 17

[78] D. Donoho. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017. 3

[79] R. Durrett. Some features of the spread of epidemics and information on a random graph. *Proceedings National Academy of Sciences*, 107(10):4491–4498, 2010. 127

[80] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17: 449–467, 1965. 9

[81] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Combinatorial Optimization Eureka, You Shrink!*, 19(2):248–264, 1972. 10

[82] W. Ellens and R. E. Kooij. Graph measures and network robustness. *CoRR*, abs/1311.5064, 2013. 124

[83] W. Elmaghraby and P. Keskinocak. Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management science*, 49(10):1287–1309, 2003. 26

[84] R. Farahani and M. Hekmatfar. Facility location: concepts, models, algorithms and case studies. *Springer Science & Business Media*, 2009. 3, 16, 21, 26, 49, 50

[85] M. C. S. Felice, D. P. Williamson, and O. Lee. A Randomized $O(\log n)$-Competitive Algorithm for the Online Connected Facility Location Problem. *Algorithmica*, 76(4):1139–1157, 2016. 22

[86] L. Ford and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. 8, 9

[87] D. Fotakis and C. Tzamos. Strategyproof facility location for concave cost functions. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 435–452. 2013. 22

[88] L. Foti, J. Lin, and O. Wolfson. Optimum versus nash-equilibrium in taxi ridesharing. *GeoInformatica*, pages 1–29, 2019. 41, 78

[89] C. Frey, A. Zufle, T. Emrich, and M. Renz. Efficient information flow maximization in probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):880–894, 2018. 127

[90] S. Fujishige, T. Hayashi, and S. Isotani. The minimum-norm-point algorithm applied to submodular function minimization and linear programming. `http://www.kurims.kyoto-u.ac.jp/preprint/file/RIMS1571.pdf`, 2006. Online; accessed 7 February 2020. 17

[91] S. Funke, A. Nusser, and S. Storandt. Placement of loading stations for electric vehicles: No detours necessary! *Journal of Artificial Intelligence Research*, 53: 633–658, 2015. 22, 25, 26

[92] B. Gamlath, S. Kale, and O. Svensson. Beating greedy for stochastic bipartite matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2841–2854. 2019. 17

[93] A. Gandini. The rise of coworking spaces: A literature review. *Ephemera : Theory and Politics in Organization*, 15(1):193–205, 2015. 71

[94] J. Garg and S. Taki. An improved approximation algorithm for maximin shares. *CoRR*, abs/1903.00029, 2019. 81

[95] R. Geisberger, D. Luxen, S. Neubauer, P. Sanders, and L. Volker. Fast detour computation for ride sharing. *arXiv preprint arXiv:0907.5269*, 0907.5269:5. 25

[96] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3): 388–404, 2012. 25

[97] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev. Multi-agent real time scheduling system for taxi companies. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 29–36, 2009. 81

[98] G. A. Godfrey and W. B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, I: single period travel times. *Transportation Science*, 36(1):21–39, 2002. 80

[99] G. A. Godfrey and W. B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, ii: Multiperiod travel times. *Transportation Science*, 36(1):40–54, 2002. 18

[100] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988. 11

[101] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990. 11

[102] A. V. Goldberg, C. Harrelson, H. Kapla, and R. F. Werneck. Efficient point-to-point shortest path algorithms. `https://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf`, 2005. Online; accessed 20 January 2019. 24

[103] J. Goldenberg, B. Libai, and E. Muller. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters*, 12(3):211–223. 101, 113

[104] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. *ACM Transactions on Knowledge Discovery from Data*, 5(4):1–37, 2012. 103

[105] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 241–250. 2010. 113, 118

[106] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems*, pages 1523–1530, 2001. 80

[107] N. Gülpinar, D. Pachamanova, and E. Çanakoglu. Robust strategies for facility location under uncertainty. *European Journal of Operational Research*, 225(1): 21–35, 2013. 42

[108] B. Gurley. A deeper look at ubers dynamic pricing model. `http://abovethecrowd.com/2014/03/11/a-deeper-look-at-ubers-dynamic-pricing-model/`, 2014. Online; accessed 28 January 2019. 27

[109] I. Gurobi Optimization. Gurobi Optimizer Reference Manual. `http://www.gurobi.com`. 49, 62

[110] Y. Hayashi and N. Uchiyama. Onion-like networks are both robust and resilient. *Scientific Reports*, 8(1):11241, 2018. 127, 150

[111] E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016. 33

[112] X. He. personal communication, 2019. 139

[113] X. He and D. Kempe. Stability of influence maximization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1256–1265, 2014. 129

[114] X. He and D. Kempe. Robust influence maximization. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 885–894, 2016. 41, 42, 124, 125, 127, 129, 130, 131, 133, 134, 139, 141

[115] X. He, G. Song, W. Chen, and Q. Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 463–474, 2012. 129, 130

[116] A. J. Hey, S. Tansley, and K. M. Tolle. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009. 3

[117] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018. 90

[118] A. Hoffman and H. Markowitz. A note on shortest path, assignment, and transportation problems. *Naval Research Logistics Quarterly*, 10(1):375–379, 1963. 10

[119] J. Holler, R. Vuorio, T. Jin, S. Singh, Z. Qin, J. Ye, X. Tang, Y. Jiao, and C. Wang. Deep reinforcement learning for dynamic multi-driver dispatching and repositioning problem. In *Proceedings of the IEEE International Conference on Data Mining*, 2019. 77, 78, 80

[120] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):26107, 2002. 108, 119, 140

[121] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, and Z. He. Top-k most influential locations selection. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2377–2380. 22

[122] X. Huang and P. Lu. An algorithmic framework for approximating maximin share allocation of chores. *CoRR*, abs/1907.04505v1, 2019. 81

[123] S. Ivanov and P. Karras. Harvester: Influence optimization in symmetric interaction networks. In *Proceedings of the IEEE International Conference Data Science and Advanced Analytics*, pages 61–70, 2016. 126

[124] S. Ivanov, K. Theocharidis, M. Terrovitis, and P. Karras. Content recommendation for viral social influence. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 565–574, 2017. 114

[125] R. K. Iyer and J. A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013. 17

[126] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34 Annual ACM Symposium on Theory of Computing*, pages 731–740, 2002. 12

[127] K. Jansen. An EPTAS for scheduling jobs on uniform processors: Using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24(2):457–485, 2010. 81

[128] S. Jegelka and A. Krause. Tutorial on submodularity in machine learning and computer vision. `https://las.inf.ethz.ch/submodularity/submodularity-2012.pdf`, 2012. Online; accessed 7 February 2020. 16, 17

[129] S. Jegelka, H. Lin, and J. A. Bilmes. On fast approximate submodular minimization. In *Advances in Neural Information Processing Systems*, pages 460–468, 2011. 17

[130] S. D. Jena, J.-F. Cordeau, and B. Gendron. Dynamic facility location with generalized modular capacities. *Transportation Science*, 49(3):484–499, 2015. 22

[131] C. Jiang, W. Li, Q. Bai, and M. Zhang. Preference aware influence maximization. In *Multi-agent and Complex Systems*, pages 153–164. 2017. 38

[132] W. Jiang and L. Zhang. The impact of the transportation network companies on the taxi industry: Evidence from beijing's GPS taxi trajectory data. *IEEE Access*, 6:12438–12450, 2018. 77

[133] J. Jin, M. Zhou, W. Zhang, M. Li, Z. Guo, Z. Qin, Y. Jiao, X. Tang, C. Wang, J. Wang, G. Wu, and J. Ye. Coride: Joint order dispatching and fleet management for multi-scale ride-hailing platforms. *CoRR*, abs/1905.11353, 2019. 28, 77, 78, 80

[134] N. Jozefowiez, F. Semet, and E.-G. Talbi. The bi-objective covering tour problem. *Computers & operations research*, 34(7):1929–1942. 22

[135] D. Kalimeris, G. Kaplun, and Y. Singer. Robust influence maximization for hyperparametric models. In *ICML*, pages 3192–3200, 2019. 129

[136] I. Kamel and C. Faloutsos. Hilbert R-tree: An Improved R-tree Using Fractals. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 500–509. 1994. 23, 63

[137] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. 131

[138] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003. 16, 37, 101, 113, 120, 127, 128, 129

[139] Y. Khorramzadeh. *Network Reliability: Theory, Estimation, and Applications*. PhD thesis, Virginia Tech, 2015. 127

[140] S. Khot and A. K. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 204–217. Springer, 2007. 81

[141] I. Y. Kim and O. De Weck. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and multidisciplinary optimization*, 31(2):105–116, 2006. 3

[142] M. Kimura, K. Saito, and H. Motoda. Blocking links to minimize contamination spread in a social network. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(2):9. 16

[143] G. W. Klau and R. Weiskircher. Robustness and resilience. In *Network Analysis*, pages 417–437. Springer Berlin Heidelberg, 2005. 125

[144] D. E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley Professional, 2009. 136

[145] D. E. Knuth. *The Art of Computer Programming, Volume 3: Retrieval on Secondary Keys*. Addison-Wesley Professional, 2009. 40

[146] M. R. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Algorithms*, 37(1):146–188, 2000. 49

[147] A. Krause and C. Guestrin. *Optimal nonmyopic value of information in graphical models: efficient algorithms and theoretical limits*. Carnegie Mellon University. Center for Automated Learning and Discovery, 2005. 16

[148] A. Krause and C. Guestrin. Beyond convexity: Submodularity in machine learning. *ICML Tutorials*, 2008. 16

[149] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 7

[150] M. Kümmel, F. Busch, and D. Z. W. Wang. Taxi dispatching and stable marriage. In *Proceedings of the 7th International Conference on Ambient Systems, Networks and Technologies*, pages 163–170, 2016. 78

[151] J. Kunegis. KONECT: the koblenz network collection. In *WWW Conference*, pages 1343–1350, 2013. 141

[152] R. Lage, P. Dolog, and M. Leginus. Vector space models for the classification of short messages on social network services. pages 209–224, 2013. 110, 117, 118

[153] P. Lahoti, K. P. Gummadi, and G. Weikum. iFair: Learning individually fair data representations for algorithmic decision making. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 1334–1345, 2019. 82

[154] E. Lam and P. V. Hentenryck. A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints*, 21(3):394–412. 22

[155] T. Lengauer and R. E. Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(1):121–141, 1979. 102

[156] J. Leskovec and R. Sosič. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016. 23, 110

[157] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007. 141

[158] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, 2007. 136

[159] N. S. Lesmana, X. Zhang, and X. Bei. Balancing efficiency and fairness in on-demand ridesourcing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 5310–5320, 2019. 41, 79, 81

[160] M. Ley and M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*, pages 1–10. 2002. 141

[161] S. Li. Approximating capacitated k-median with $(1+\epsilon)$ k open facilities. In *Proceedings of the 27 annual ACM-SIAM symposium on Discrete algorithms*, pages 786–796. 2016. 49

[162] S. Li. Private Communication, 2018. 49

[163] S. Li and S. Li. On uniform capacitated k-median beyond the natural lp relaxation. *ACM Transactions on Algorithms (TALG)*, 13(2):1–18, 2017. 49, 50, 62

[164] X. Li, J. Ma, J. Cui, A. Ghiasi, and F. Zhou. Design framework of large-scale one-way electric vehicle sharing systems: A continuum approximation model. *Transportation Research Part B: Methodological*, 88:21–45, 6 2016. 22

[165] Y. Li, J. Fan, Y. Wang, and K. Tan. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10): 1852–1872, 2018. 19, 38, 39, 100, 101, 113, 124, 127, 129

[166] S. Lim, J. Shin, N. Kwak, and K. Jung. Phase transitions for information diffusion in random clustered networks. *The European Physical Journal B*, 89 (9):188, 2016. 127

[167] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783. 2018. 28, 34, 77, 78, 80, 88, 89, 91

[168] J. Liu, M. Zhou, S. Wang, and P. Liu. A comparative study of network robustness measures. *Frontiers of Computer Science*, 11(4):568–584, 2017. 124

[169] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. Mining topic-level influence in heterogeneous networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 199–208, 2010. 114

[170] A. Logins. Data-driven algorithms for spatial networks flow problems. Master's thesis, Skolkovo Institute of Science and Technology, 2016. 11

[171] A. Logins and P. Karras. An experimental study on network immunization. In *Proceedings of the 23rd International Conference on Extending Database Technology*, pages 726–729, 2019. 100, 131, 141, 150

[172] A. Logins and P. Karras. Content-based network influence probabilities: Extraction and application. In *2019 International Conference on Data Mining Workshops*, pages 69–72. 2019. 100

[173] A. Logins, P. Karras, and C. S. Jensen. Multicapacity facility selection in networks. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 794–805, 2019. 47

[174] A. Logins, Y. Li, and P. Karras. On the robustness of cascade diffusion under node attacks. In *Proceedings of the Web Conference*, 2020. 123

[175] A. Logins, Y. Li, and P. Karras. On the robustness of cascade diffusion under node attacks. *Submitted to IEEE Transactions on Knowledge and Data Engineering*, 2020. 124

[176] A. Logins, L. H. U, and P. Karras. Fair cruising. In *Submitted to the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020. 77

[177] O. Lordan and M. Albareda-Sambola. Exact calculation of network robustness. *Reliability Engineering & System Safety*, 183:276–280, 2019. 126, 132

[178] L. A. Lorena and E. L. Senne. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876, 2004. 48, 50

[179] L. Lovász and M. Plummer. *Matching Theory*. Elsevier Science, 1986. 7, 9, 11, 12, 13, 15

[180] L. Lovasz, M. Grotschel, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988. 14, 17

[181] W. Lu and L. Quadrifoglio. Fair cost allocation for ridesharing services - modeling, mathematical programming and an algorithm to find the nucleolus. *CoRR*, abs/1902.07266, 2019. 78

[182] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*, volume 228. Springer, 2015. 13

[183] N. Marković, I. O. Ryzhov, and P. Schonfeld. Evasive flow capture: A multi-period stochastic facility location problem with independent demand. *European Journal of Operational Research*, 257(2):687–703, 2017. 22

[184] M. Marufuzzaman, R. Gedik, and M. S. Roni. A Benders based rolling horizon algorithm for a dynamic facility location problem. *Computers & Industrial Engineering*, 98:462–469. 22

[185] P. Massa, M. Salvetti, and D. Tomasoni. Bowling alone and trust decline in social network sites. In *Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 658–663, 2009. 141

[186] A. Maulana, M. Kefalas, and M. T. M. Emmerich. Immunization of networks using genetic algorithms and multiobjective metaheuristics. In *IEEE Symposium Series on Computational Intelligence*, pages 1–8, 2017. 103

[187] R. P. McAfee and V. Te Velde. Dynamic pricing in the airline industry. *Handbook on Economics and Information Systems*, 2006. 26, 27

[188] S. Mitra. Identifying top-K optimal locations for placement of largescale trajectory-aware services. 2016. 21, 22, 63

[189] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. 89

[190] F. Morone and H. A. Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524(7563):65, 2015. 128

[191] K. Nagano and Y. Kawahara. Structured convex optimization under submodular constraints. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. 2013. 17

[192] V. Nanda, P. Xu, K. A. Sankararaman, J. P. Dickerson, and A. Srinivasan. Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours. In *AAAI/ACM Conference on AI, Ethics, and Society*, page 131, 2020. 33, 78

[193] M. Ndiaye and H. Alfares. Modeling health care facility location for moving population groups. *Computers & Operations Research*, 35(7):2154–2161. 22

[194] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *SIGMETRICS*, pages 211–222, 2012. 113

[195] B. Neuberg. `http://coworking.com`, 2020. 71

[196] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. *CoRR*, abs/1812.11794, 2018. 34, 80

[197] Y. V. Nikulin. Robustness in combinatorial optimization and scheduling theory: An annotated bibliography. Technical report, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 2004. 42

[198] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Dynamic influence analysis in evolving networks. *PVLDB*, 9(12):1077–1088, 2016. 133, 134, 138, 139, 140

[199] A. Pagani, G. Mosquera, A. Alturki, S. Johnson, S. Jarvis, A. Wilson, W. Guo, and L. Varga. Resilience or robustness: identifying topological vulnerabilities in rail networks. *Royal Society Open Science*, 6(2):181301, 2019. 124

[200] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 131

[201] S. Pallottino and M. G. Scutella. Shortest path algorithms in transportation models: classical and innovative aspects. In *Equilibrium and advanced transportation modelling*, pages 245–281. Springer, 1998. 25

[202] M. Pan, Y. Li, X. Zhou, Z. Liu, R. Song, H. Lu, and J. Luo. *Dissecting the Learning Curve of Taxi Drivers: A Data-Driven Approach*, pages 783–791. 2019. 78, 79, 80

[203] S. Park, T.-E. Lee, and C. S. Sung. A three-level supply chain network design model with risk-pooling and lead times. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):563–581, 2010. 22

[204] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017. 35

[205] V. H. L. Patricio, F. Daolio, H. J. Herrmann, and M. Tomassini. *Propagation Phenomena in Real World Networks. Generating Robust and Efficient Networks Under Targeted Attacks*, pages 215–224. Intelligent Systems Reference Library. Springer, 2015. 124

[206] G. Paul, T. Tanizawa, S. Havlin, and H. E. Stanley. Optimization of robustness of complex networks. *The European Physical Journal B*, 48(1):149–149, 2005. 126

[207] S. A. Pedersen, B. Yang, and C. S. Jensen. Fast stochastic routing under time-varying uncertainty. *The VLDB Journal*, Oct 2019. 25

[208] R. L. Phillips. *Pricing and revenue optimization*. Stanford University Press, 2005. 26

[209] D. V. Phung, T. M. Hoang, A. T. Nguyen, and T. B. Dinh. Dta hunter system: A new statistic-based framework of predicting future demand for taxi drivers. In *Proceedings of the 8th International Symposium on Information and Communication Technology*, pages 159–166, 2017. 78

[210] W. B. Powell and H. Topaloglu. Approximate dynamic programming for large-scale resource allocation problems. In *Models, Methods, and Applications for Innovative Decision Making*, pages 123–147. INFORMS, 2006. 18

[211] S. Qian, J. Cao, F. L. Mouël, I. Sahel, and M. Li. SCRAM: A sharing considered route assignment mechanism for fair taxi route recommendations. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964, 2015. 78

[212] G. Qin, T. Li, B. Yu, Y. Wang, Z. Huang, and J. Sun. Mining factors affecting taxi drivers incomes using gps trajectories. *Transportation Research Part C: Emerging Technologies*, 79:103 – 118, 2017. 42, 78, 79

[213] Z. Qu, Z. Pan, Y. Chen, X. Wang, and H. Li. A distributed control method for urban networks using multi-agent reinforcement learning based on regional mixed strategy nash-equilibrium. *IEEE Access*, 2020. 41

[214] S. S. Reynolds and D. Rietzke. Price caps, oligopoly, and entry. *Economic Theory*, 66(3):707–745, Oct 2018. 29

[215] J. M. Robertson and W. A. Webb. *Cake-cutting algorithms - be fair if you can.* A K Peters, 1998. 42, 81

[216] H. Rock. Scaling techniques for minimal cost network flows. *Discrete structures and algorithms*, 1980. 11

[217] J. Rodriguez. The science behind openai five that just produced one of the greatest breakthrough in the history of AI. `https://bit.ly/31G8T7z`, 2018. Online; accessed 30 January 2019. 35

[218] I. Rodriguez-Martin and J. J. Salazar-Gonzalez. Solving a capacitated hub location problem. *European Journal of Operational Research*, 184(2):468–479, 2008. 22

[219] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu. The rich and the poor: A markov decision process approach to optimizing taxi driver revenue efficiency. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2329–2334, 2016. 78, 79

[220] S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009. 22

[221] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, pages 4292–4293, 2015. 141

[222] K. J. Rothman, S. Greenland, and T. L. Lash. *Modern Epidemiology*. Lippincott Williams & Wilki, 2013. 127

[223] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016. 19

[224] M. Rönnqvist, S. Tragantalerngsak, and J. Holt. A repeated matching heuristic for the single-source capacitated facility location problem. *European Journal of Operational Research*, 116(1):51–68, jul 1999. 12, 23

[225] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-based intelligent information and engineering systems*, pages 67–75. 2008. 113

[226] C. Saxena, M. N. Doja, and T. Ahmad. Group based centrality for immunization of complex networks. *Physica A*, 508:35–47, 2018. 103

[227] K. Scaman, A. Kalogeratos, L. Corinzia, and N. Vayatis. A spectral method for activity shaping in continuous-time information cascades. *CoRR*, abs/1709.05231, 2017. 40, 101, 103, 107, 110, 131

[228] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005. 117, 141

[229] T. Schieber, M. Ravetti, and P. M. Pardalos. A review on network robustness from an information theory perspective. In *Proceedings of the International Conference on Discrete Optimization and Operations Research*, pages 50–60. 2016. 124

[230] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences*, 108(10):3838–3841, 2011. 124, 126, 149

[231] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 90

[232] N. Schwartz, R. Cohen, D. ben Avraham, A.-L. Barabási, and S. Havlin. Percolation in directed scale-free networks. *Physical Review E*, 66(1):015104, 2002. 132

[233] I. Segalovich. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications*, pages 273–280, 2003. 117

[234] P. Shakarian, A. Bhatnagar, A. Aleali, E. Shaabani, and R. Guo. The independent cascade and linear threshold models. In *Diffusion in Social Networks*, pages 35–48. Springer, 2015. 40, 101, 103, 113

[235] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274. 1997. 22

[236] L. V. Snyder. Facility location under uncertainty: a review. *IIE transactions*, 38(7):547–564, 2006. 24, 26, 42

[237] L. V. Snyder, M. S. Daskin, and C.-P. Teo. The stochastic location model with risk pooling. *European Journal of Operational Research*, 179(3):1221–1238, 2007. 22

[238] Y. Song and T. N. Dinh. Optimal Containment of Misinformation in Social Media: A Scenario-Based Approach. In *International Conference on Combinatorial Optimization and Applications*, pages 547–556. 2014. 39

[239] M. Staib and S. Jegelka. Robust budget allocation via continuous submodular functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3230–3240, 2017. 3, 18, 42

[240] D. Stauffer and A. Aharony. *Introduction to percolation theory*. Taylor & Francis, 2003. 124, 126

[241] Y. Su and X. Yan. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246, 2017. 118

[242] K. Subbian, C. Aggarwal, and J. Srivastava. Content-centric flow mining for influence analysis in social streams. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 841–846, 2013. 114

[243] T. Sühr, A. J. Biega, M. Zehlike, K. P. Gummadi, and A. Chakraborty. Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3082–3092, 2019. 42, 79

[244] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 33, 34

[245] H. Taha. *Operations Research an Introduction*. Pearson, 2017. 3

[246] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554. 2015. 40, 120, 133, 141

[247] J. Tariq, M. Ahmad, I. Khan, and M. Shabbir. Scalable approximation algorithm for network immunization. page 200, 2017. 101, 102, 107

[248] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. 151

[249] V. Tejaswi, P. V. Bindu, and P. S. Thilagam. Diffusion models and approaches for influence maximization in social networks. pages 1345–1351, 2016. 101

[250] G. Tennenholtz, C. Caramanis, and S. Mannor. The stochastic firefighter problem. *CoRR*, abs/1711.08237, 2017. 103

[251] G. Tong, W. Wu, S. Tang, and D.-Z. Du. Adaptive Influence Maximization in Dynamic Social Networks. *IEEE/ACM Transactions on Networking*, 25(1): 112–125, 2017. 38

[252] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In *Proceedings of the 2018 International Conference on Management of Data*, pages 773–788, 2018. 27, 28

[253] C. Tseng and C. Chau. Viability analysis of electric taxis using new york city dataset. In *Proceedings of the 8th International Conference on Future Energy Systems*, pages 328–333, 2017. 79

[254] S. Tsugawa and H. Ohsaki. On the robustness of influence maximization algorithms against non-adversarial perturbations. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 91–94, 2017. 129

[255] C. Tzamos and C. A. Wilkens. The value of knowing your enemy. *CoRR*, abs/1411.1379, 2014. 28

[256] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis. Capacity constrained assignment in spatial databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 15–28, 2008. 52, 54

[257] L. H. U, K. Mouratidis, M. L. Yiu, and N. Mamoulis. Optimal matching between spatial datasets under capacity constraints. *ACM transactions on database systems*, 35(2):9:1–9:44, 2010. 10, 11, 52, 54, 59, 60, 61

[258] Z. Ulukan and E. Demircioglu. A survey of discrete facility location problems. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 9(7), 2015. 16, 21, 23

[259] Unity. Solving sparse-reward tasks with Curiosity. `https://blogs.unity3d.com/2018/06/26/solving-sparse-reward-tasks-with-curiosity/`, 2018. Online; accessed 30 January 2019. 35

[260] J. Von Neumann. On the theory of games of strategy. *Contributions to the Theory of Games*, 4:13–42, 1959. 41

[261] E. Vynnycky and R. White. *An introduction to infectious disease modelling*. Oxford University Press, 2010. 125

[262] B. Wang, G. Chen, L. Fu, L. Song, and X. Wang. DRIMUX: Dynamic rumor influence minimization with user experience in social networks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 29(10):2168–2181, 2017. 103, 110, 126

[263] H. Wang and H. Yang. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, 129:122 – 155, 2019. 77

[264] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. Technical report, 2003. 102, 110

[265] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 1478–1489, 2019. 33

[266] Z. Wang, C. Chen, and W. Li. Information diffusion prediction with network regularized role-based user representation learning. *ACM Trans. Knowl. Discov. Data*, 13(3):29:1–29:23, May 2019. 115

[267] D. J. Watts and S. H. Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 393(6684):440, 1998. 108, 140

[268] R. Waury, C. S. Jensen, S. Koide, Y. Ishikawa, and C. Xiao. Indexing trajectories for travel-time histogram retrieval. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology*, pages 157–168, 2019. 25

[269] J. K. Weber and V. S. Pande. Percolation-like phase transitions in network models of protein dynamics. *The Journal of Chemical Physics*, 142(21):215105, 2015. 126

[270] E. W. Weisstein. Monte Carlo Method. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/MonteCarloMethod.html`, 2020. Online; accessed 30 January 2019. 18

[271] A. W. Wijayanto and T. Murata. Pre-emptive spectral graph protection strategies on multiplex social networks. *Applied Network Science*, 3(1):5. 103, 110

[272] B. Wilder, . S.-C. Suen, . M. Tambe, D. of Computer, S. 2Department, of Industrial, and S. Engineering. Preventing infectious disease in dynamic populations under uncertainty. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 841–848, 2018. 103

[273] T.-H. Wu and J.-N. Lin. Solving the competitive discretionary service facility location problem. *European Journal of Operational Research*, 144(2):366–378, 2003. 22

[274] F. Xiao, J. X. Zhou, and Z. a Hu. Location and pricing of competitive service providers for post-disaster relief. *International Journal of Industrial and Systems Engineering*, 25(1):14, 2017. 42

[275] Y. Yamaguchi and T. Maehara. Stochastic packing integer programs with few queries. In A. Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 293–310. 2018. 17

[276] D. Yan, J. Cheng, Z. Zhao, and W. Ng. Efficient location-based search of trajectories with location importance. *Knowledge and Information Systems*, 45 (1):215–245, 2015. 22

[277] D. Yang, X. Liao, H. Shen, X. Cheng, and G. Chen. Relative influence maximization in competitive social networks. *Science China Information Sciences*, 60(10):108101, 2017. 38

[278] D. Yang, X. Liao, H. Shen, X. Cheng, and G. Chen. Dynamic node immunization for restraint of harmful information diffusion in social networks. *Physica A*, 503:640–649, 2018. 101, 103, 110

[279] Y. Yang, E. Chen, Q. Liu, B. Xiang, T. Xu, and S. A. Shad. On approximation of real-world influence spread. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 548–564. 2012. 39

[280] B. Yao, X. Xiao, F. Li, and Y. Wu. Dynamic monitoring of optimal locations in road network databases. *The VLDB Journal*, 23(5):697–720, 2014. 22, 23, 51

[281] H. Yildirim, V. Chaoji, and M. J. Zaki. DAGGER: A scalable index for reachability queries in large dynamic graphs. *CoRR*, abs/1301.0977, 2013. 125, 135, 138, 151

[282] E. Yilmaz, S. Elbasi, and H. Ferhatosmanoglu. Predicting Optimal Facility Location without Customer Locations. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2121–2130. 2017. 40, 51, 71, 72

[283] X. Yu, S. Gao, X. Hu, and H. Park. A Markov decision process approach to vacant taxi routing with e-hailing. *Transportation Research Part B: Methodological*, 121(C):114–134, 2019. 79

[284] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates. FA*IR: A Fair Top-k Ranking Algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578, 2017. 82

[285] C. Zhang and J. A. Shah. Fairness in multi-agent sequential decision-making. In *Proceedings in the Annual Conference on Neural Information Processing Systems*, pages 2636–2644, 2014. 80

[286] C. Zhang and J. A. Shah. On fairness in decision-making under uncertainty: Definitions, computation, and comparison. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3642–3648, 2015. 43

[287] H. Zhang, S. Mishra, M. T. Thai, J. Wu, and Y. Wang. Recent advances in information diffusion and influence maximization in complex social networks. *Opportunistic Mobile Social Networks*, 37(1.1):37, 2014. 127

[288] J. Zhang, J. Tang, J. Li, Y. Liu, and C. Xing. Who influenced you? predicting retweet via social influence locality. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):25:1–25:26, 2015. 113

[289] R. Zhang and R. Ghanem. Demand, supply, and performance of street-hail taxi. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2019. 78

[290] Y. Zhang. *Optimizing and Understanding Network Structure for Diffusion*. PhD thesis, Virginia Tech. 101

[291] Y. Zhang and B. A. Prakash. Data-aware vaccine allocation over large networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(2):20:1–20:32, 2015. 101, 102, 103, 107, 110, 112, 113, 119, 120, 127, 130, 131

[292] Y. Zhang, A. Ramanathan, A. Vullikanti, L. L. Pullum, and B. A. Prakash. Data-driven immunization. In *Proceedings of the IEEE International Conference on Data Mining*, pages 615–624, 2017. 104, 110

[293] Y. Zhao, I. Borovikov, J. Rupert, C. Somers, and A. Beirami. On multi-agent learning in team sports games. *CoRR*, abs/1906.10124, 2019. 34, 35

[294] L. Zheng, P. Cheng, and L. Chen. Auction-based order dispatch and pricing in ridesharing. In *Proceedings of the IEEE 35th International Conference on Data Engineering*, pages 1034–1045. 2019. 28

[295] M. Zhou, J. Jin, W. Zhang, Z. Qin, Y. Jiao, C. Wang, G. Wu, Y. Yu, and J. Ye. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2645–2653. 2019. 28, 77, 78, 80

[296] A. Zockaie, H. Z. Aashtiani, and M. Ghamami. Solving detourbased fuel stations location problems. *ComputerAided Civil and Infrastructure Engineering*, 31(2):132–144, 2016. 22

[297] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018. 42, 43