Theoretical and Practical Aspects of Finite Closure Systems for Mining and Learning

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt
von

Florian Seiffarth

aus
Kreuztal

Bonn, 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter: Prof. Dr. Stefan Wrobel
Gutachter: Prof. Dr. Christian Bauckhage

Tag der Promotion: 14.06.2023
Erscheinungsjahr: 2023

Declaration I, Florian Seiffarth, confirm that this work is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at the point of their use. A full list of the references employed has been included.

Acknowledgments A very special thank goes to my supervisors Tamás Horváth and Stefan Wrobel, without whom this work would not have been possible. I am very grateful to have received such intensive supervision. It was always something special to be invited by Tamás to Nemesvita, a small village near Lake Balaton, and to work there in peace on my current research problems. These visits have always inspired and motivated me. Not only new ideas and some of our joint papers were created here, but thanks to Tamás, I also became an expert in the culinary offer in the surroundings of Nemesvita.

The MLAI group has grown a lot during my time at the institute. What always has remained is the great atmosphere in our group, which has helped me in my research and writing this thesis. I have enjoyed all the coffee breaks, darts games, adventurous hikes, and joint conferences. Thank you, Fouad Alkhoury, Ewald Bindereif, Michael Kamp, Sebastian Müller, Till Schulz, Patrick Seifner, Eike Stadtländer, Vanessa Toborek, Daniel Trabold and Pascal Welke. I would like to thank Tamás Horváth, Daniel Trabold, and Pascal Welke for their feedback on the early versions of my thesis and Victor Chepoi for his valuable comments on an early version of a paper.

Last but not least, I would like to thank my parents, Antje and Frank Seiffarth, and my wife, Tatjana Seiffarth, for their tremendous support. During the last period of my thesis I was very motivated and excited about my son, who will be born soon (hopefully *after* I hand in this thesis).

Abstract This thesis investigates the potential and limitations of different adaptations of *half-space separations* in ordinary Euclidean spaces, one of the most popular paradigms in machine learning, to abstract finite *closure systems*. Although closure systems and their corresponding closure operators have different applications in *machine learning* and *data mining*, this question has not been studied in these research fields. Furthermore, all machine learning and data mining applications of closure systems have so far been restricted to *specific* domains.

Using Jamison's notion of half-spaces in abstract closure systems, for the adaptation we regard a closed set as a half-space if its complement is also closed. However, this generalization does not preserve the most important properties of ordinary half-space separations. In particular, there is no result corresponding to Minkowski's hyperplane separation theorem for Euclidean spaces because disjoint closed sets are not necessarily separable by half-spaces in abstract closure systems. We show that deciding the *half-space separation* (HSS) problem, that is, the problem whether or not two disjoint closed sets are half-space separable in an abstract closure system is NP-hard in general. Motivated by this negative complexity result, in a first step we therefore focus on the class of closure systems in which the HSS problem has a solution for *any* two disjoint closed sets. This kind of closure systems will be referred to as the *Kakutani* closure systems. On the one hand, for the case that we have access to the abstract finite closure system via the underlying closure operator only, we show the negative worst-case result that one needs exponentially many closure operator calls to decide whether the closure system is Kakutani, or not. On the other hand, we give a forbidden minor characterization of *Kakutani* geodesic closure systems over graphs, which, for example, implies that disjoint closed sets in outerplanar graphs are always separable via half-spaces. As a second direction to overcome the above negative result, we relax the HSS problem to finding two *inclusion maximal* disjoint closed sets and present a simple greedy algorithm for this problem. It turns out that this simple algorithm has several attractive properties. In particular, it is optimal with respect to the number of closure operator calls, provides an algorithmic characterization of Kakutani closure systems, and can be extended to returning *maximum margin* separations by utilizing the notion of monotone linkage functions.

Regarding some practical aspects of the generic theory developed in the thesis, we demonstrate on real-world networks up to more than 100,000,000 edges that their geodesic cores can closely be approximated. As a practical application, we empirically show that the Tukey depth of the vertices in a graph can closely be approximated by our greedy algorithm.

Contents

1 Introduction

Linear separations in Euclidean spaces, that is, separating positive and negative examples in $\mathbb{R}^d$ by some hyperplane has been at the heart of *machine learning* for a long time. The class label of an unseen data point can be predicted in this way by that of the half-space containing it. A common problem in this field of research is to work with non-Euclidean instance spaces. Kernel methods provide an elegant solution to this problem by (implicitly) embedding the data into some inner product feature space, which, however, is not necessarily known to the designer of the kernel function. Motivated by the fact that half-spaces are *convex*, and hence, *closed* in an algebraic sense, in this thesis we investigate an alternative way by studying the *potential* and *limitations* of *adapting* the successful paradigm of linear separations in Euclidean spaces to abstract *closure systems*.

Closure systems and algebraic *closure operators* (see, e.g., Davey and Priestley, 2002) play a central role not only in universal algebra, but also in different fields of computer science, in particular, in *machine learning* and *data mining* (Witten et al., 2011). A closure system over a ground set E is a family $\mathcal{C}$ of subsets of E such that $E \in \mathcal{C}$ and the intersection of every non-empty subset of $\mathcal{C}$ is again an element of $\mathcal{C}$. The origins of closure systems date back to the research of the French mathematician Évariste Galois in the early 19th century. His *Galois theory* is directly related to closure systems considering the so-called *Galois connections* introduced by Ore (1944). Although the notion of closure operators appeared implicitly in the works of Ernst Schröder, Richard Dedekind, or Georg Cantor, it was formally introduced independently by Eliakim Hastings Moore and Frigyes Riesz. In particular, while Riesz (1909) studied topological aspects of closure systems, Moore (1910) investigated the origins of closure operators. The primary goal of all these abstractions was to axiomatize the properties of *convex hulls* in $\mathbb{R}^d$. This axiomatic characterization of convex hulls was subsequently used successfully in various fields of mathematics, including, for example, topology (Kuratowski, 1922), logic (Tarski, 1942), lattice theory (Birkhoff, 1940; Stone, 1938; Ward, 1942), discrete mathematics (Korte et al., 2012), and computational geometry (van de Vel, 1993).

An essential property of closure systems is that they can be characterized in terms of *closure operators*. This characterization is necessary for *algorithmic* applications of closure systems; otherwise, their *extensional representations* may cause severe complexity problems in computer science applications. In particular, even in the case of finite domains, the number of closed sets can be *exponential* in the cardinality of the ground set at hand. In contrast, their *intensional representations* via closure operators may allow for efficient utilization of closure systems in computer science. The concept of closure systems has been used and studied in different areas of computer science. As a first example, consider the field of *relational databases*. The theory of relational databases (see, e.g., Codd, 1970) is concerned with implications and implicational bases, where each implicational basis gives rise to a closure system. In particular, two implicational bases are called equivalent if they define the same closure system. In this application context, it is a common question to find minimum and optimal bases for closure systems (see, e.g., Adaricheva et al., 2013). As a second example, first-order Horn clauses, used, e.g., in logic programming (Lloyd, 2012) provide another, at first sight not obvious representation of closure systems. Hence, closure systems are directly connected to applications in predicate logic (Barwise, 1977). Besides these examples, closure systems are also related, e.g., to matroids and greedoids (Korte et al., 2012), in discrete mathematics.

We now turn to applications of closure systems in *machine learning* and *data mining*. We first mention some standard applications of closure systems in *local pattern mining* a subfield of data mining[1]. In particular, Galois connections are central to *itemset mining* (Pasquier et al., 1999), by noting that several results obtained for closed itemset mining apply almost directly to *formal concept analysis* (Ganter et al., 2005) as well. We regard databases of transactions in itemset mining or formal contexts in formal concept analysis as *binary* matrices. Then, one can naturally define a *lower adjoint* function mapping subsets of the columns to subsets of the rows of such a matrix and an *upper adjoint* function mapping subsets of rows to subsets of columns. It is a well-known fact (see, e.g., Davey and Priestley, 2002) that the composition of these two maps then gives rise to a closure operator on the set of columns. One of the related computational tasks is to generate *all* fixed points (or closed sets) of the closure operator obtained in this way. Several algorithms have been designed for this enumeration problem (see, e.g., Boros et al., 2003; Gély, 2005; Pasquier et al., 1999) and for more general problem settings (see, e.g., Boley et al., 2010; Ganter and Reuter, 1991).

Although a part of this work will be devoted to graph mining, the primary focus of the thesis is on *concept learning* over *arbitrary* domains, by *adapting* the notion of *linear separation* in $\mathbb{R}^d$ and in other inner product spaces. The theoretical motivation and the idea behind this adaptation lie in the advantageous algorithmic and formal properties of *ordinary* linear separations in $\mathbb{R}^d$.

For the reader's convenience, we shortly sketch the *supervised learning task* that we are interested in this work: *Given* disjoint sets of positive and negative examples that are *linearly separable*[2], *find* a hypothesis consistent with all examples, i.e., which contains all positive examples and none of the negative ones. It is a well-known fact that such learning tasks in $\mathbb{R}^d$ can be solved for example by the Perceptron algorithm introduced by Rosenblatt (1958) or in inner product feature spaces by hard margin Support Vector Machine (SVMs) invented by Vapnik and his co-authors (Boser et al., 1992). Figure 1.1 provides a comparative visual example of the difference in separation between the two

[1]While machine learning is mainly interested in extracting *global* patterns from the data at hand, data mining has its focus on generating *local* patterns only (Witten et al., 2011).

[2]By "linearly separable" we mean the original definition in $\mathbb{R}^d$ as well as its adaptation to abstract closure systems.

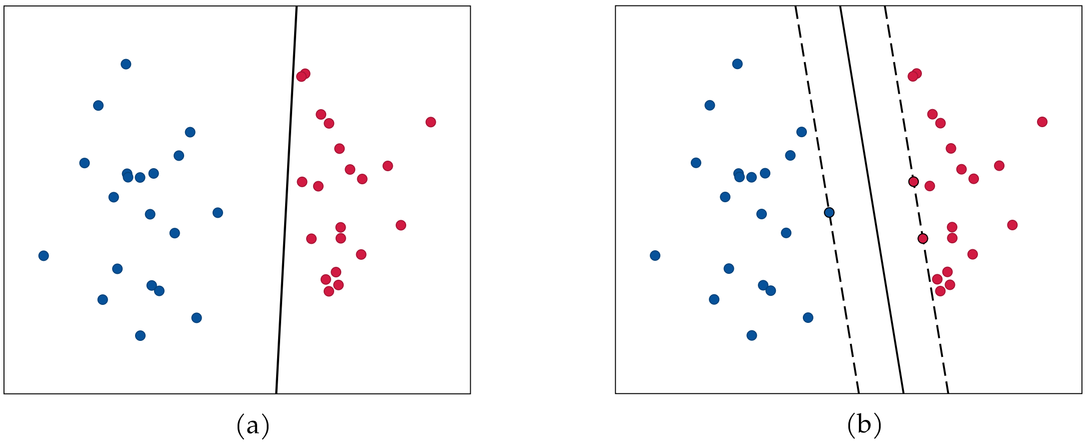


Figure 1.1: Example of linear separations in $\mathbb{R}^2$ induced by the Perceptron algorithm (a) and Support Vector Machines (b). The respective hyperplane is given in bold while in case of SVMs, the maximum margins are given by dashed lines.

algorithms. It is an elementary fact that the above algorithms are applicable if and only if the input data is *linearly separable*, i.e., there exists a hyperplane such that the two sets we want to separate lie on their opposite sides. Using an old result of Kakutani (1937), this condition can be formulated equivalently as follows: Two subsets of $\mathbb{R}^d$ (or some inner product space) can be separated by *half-spaces* if and only if their *convex hulls* are disjoint (see, also, Figures 1.2(a) and 1.2(b)).

Thus, in order to adapt linear separability in $\mathbb{R}^d$ to *arbitrary* domains, we need to generalize the notions of *convex hulls* and *half-spaces* in $\mathbb{R}^d$ to *abstract* concept classes or hypothesis spaces, i.e., set systems over some ground set (also referred to as domain or instance space). Closure systems provide such a generalization of convex hulls in a very natural way. Indeed, given a subset S of the domain, the notion of the convex hull in $\mathbb{R}^d$ naturally corresponds to the *smallest* set in the underlying closure system that contains S. Since, by definition, closure systems are intersection closed set systems, such a smallest set always exists. Regarding the generalization of half-spaces, it is natural to apply the following definition of Jamison (1974): Given a closure system $\mathcal{C}$ over some ground set E, a set $H \subseteq E$ is an *abstract half-space* if and only if H and its complement $H^c := E \setminus H$ are both closed (i.e., $H, H^c \in \mathcal{C}$).

Using these definitions of convex hulls and half-spaces, one might try to adapt, at least at first glance, Kakutani's above characterization of linear separability in $\mathbb{R}^d$ to that in abstract closure systems as follows: Two subsets of the underlying domain are linearly separable, i.e., contained by two *closed* sets partitioning the domain, if and only if their closures are disjoint. The following example, however, demonstrates that we face an entirely different situation in case of *abstract* closure systems: Consider the separation problem shown in Figure 1.2(c). The domain E consists of the eight points of $\mathbb{R}^2$ given in the figure. A subset C of E is regarded as closed if it is equal to the intersection of its convex hull in $\mathbb{R}^2$ with E. Although the closures of the three red and the three blue points

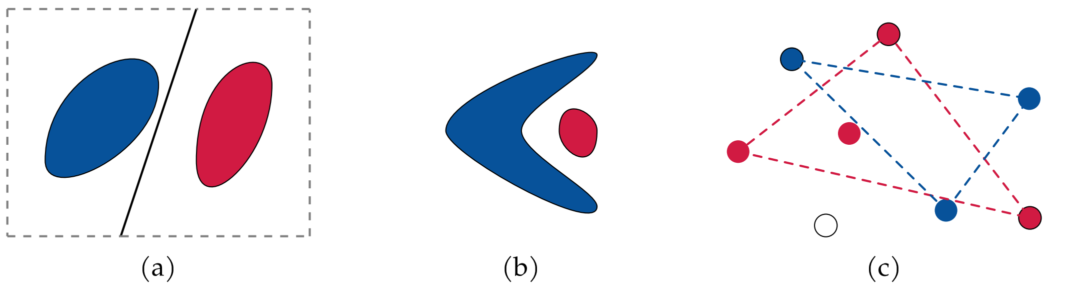


Figure 1.2: (a) Half-Space separation in $\mathbb{R}^d$ (b) non-convex input data which is not linearly separable and (c) disjoint convex sets in the finite case. The convex sets in (a) are disjoint and hence linearly separable via a hyperplane. The sets in (b) are disjoint but not separable because the blue set is non-convex and its convex hull contains the red set. The closures of the red and blue point sets in (c) are disjoint (because only the traces and *not* the enclosed areas in $\mathbb{R}^2$ are considered), but there is no complete partitioning of the space by closed sets containing the two input sets. Hence, the sets of blue and red points are *not* half-space separable.

connected by dashed lines are disjoint, they are *not* separable by abstract half-spaces in the closure system defined in this way. Indeed, there exists no closed set containing the red points and the uncolored point (resp. the blue points and the uncolored point) which is disjoint with the closed set containing the blue (resp. red) points. This and other examples raise the following research question that will be in the focus of the thesis:

To what extent is it possible to adapt linear separability in $\mathbb{R}^d$ to abstract closure systems?

We restrict this question to the case that the underlying ground set is *finite* and *motivate* the study of this research question from different viewpoints. First of all, the adaptation of linear separability in $\mathbb{R}^d$ to finite closure systems is a crucial *theoretical* question in its own right. The answers to this question may provide *new insights* into the problem and allow for distinguishing between domain-independent and domain-dependent properties utilized by the ordinary algorithms by Boser et al. (1992); Rosenblatt (1958). Regarding our second argument, recall that one of the attractive properties of kernel methods, besides their applicability to data beyond the usual representation with single tables of fixed width, is that they consist of a *domain-independent* algorithm parametrized by a *domain-dependent* kernel function. Our approach on concept learning in finite closure systems follows this paradigm. That is, it consists of a *domain-independent* algorithm parametrized by a *domain-dependent* closure operator. However, compared to kernel methods, the designers of the underlying closure operator have much more *control* over its semantics than those of kernel functions. That is, it is much *easier* to integrate domain-specific knowledge into the closure operator, compared to kernel methods (which are more or less black-box models from this point of view). As a third argument, we note that the generality of the adaptation may allow for considering linear separation on domains that are *different* from $\mathbb{R}^d$ and other inner product spaces. For example, we will give applications, where the underlying domain is the vertex set of a graph or a finite lattice. As a fourth argument, our approach provides a high "*expressive power*" for defining the particular notion of "convexity" over the domain at hand. In particular, the designer of the closure operator has the freedom to consider a broader class of closure systems containing closed sets that are not necessarily half-spaces. Last but not least, the adaptation can naturally be extended to *multi-class* learning tasks in abstract closure systems. This is because positive and negative examples are handled symmetrically in the binary case.

Before discussing the main contributions of the thesis, we note that to answer our question formulated above, one of the central technical challenges is to *synthesize* and *adapt* results from entirely different fields of computer science and mathematics. In particular, our approach relies on a careful combination of results from *machine learning* and *data mining*, *computational geometry*, *discrete mathematic*s, *graph theory*, and *lattice theory*.

1.1 Contributions: Questions and Results

In this section we provide an overview of the main results of the thesis. The contributions can be split into a *theoretical* and a *practical* part. Accordingly, we first summarize our theoretical results concerning half-space (Section 1.1.1) and maximal closed set separations (Section 1.1.2) in finite closure systems. Subsequently, we discuss two *practical* aspects of geodesic closure systems over graphs: One concerning the approximation of geodesic closed sets with applications to geodesic *core-periphery* decompositions of large real-world networks and one with an application of our maximal closed set separation algorithm to the approximation of the *graph Tukey depth* (Section 1.1.3).

1.1.1 Half-Space Separation in Finite Closure Systems

As demonstrated with the simple example in Figure 1.2(c), there exist closure systems for which the result of Kakutani (1937) does not hold. That is, the disjointness of the closures of the two input sets does not imply their half-space separability. This problem motivates our first natural question of whether it can be decided *efficiently* if two sets are half-space separable in the underlying abstract closure system, or not. More precisely, we first consider the following decision problem:

(1) Given some finite ground set E, a family of closed sets $\mathcal{C}$ over E, and $A, B \subseteq E$, does there *exist* a closed set $H \in \mathcal{C}$ such that $H^c = E \setminus H \in \mathcal{C}$, $A \subseteq H$, and $B \subseteq H^c$?

We call this decision problem the *half-space separation* (HSS) problem. Using a reduction from the NP-complete Convex 2-Partitioning problem (Artigas et al., 2011), we prove that the HSS problem is NP-hard for *arbitrary* closure systems. Furthermore, it is NP-complete for the case that the underlying closure operator can be calculated in time *polynomial* in the cardinality of the ground set.

These negative results motivate us to *relax* the problem setting. An obvious option is to *restrict* the HSS problem to such closure systems which preserve the Kakutani property.

That is, the closure system is required to satisfy the following condition: Two sets are half-space separable in the closure system if and only if their closures are disjoint. Given a finite closure system, the related natural first question is if it is possible to decide whether it fulfills the Kakutani property, or not. More precisely, our second question is concerned with the following decision problem:

(2) Given a finite closure system, is it possible to decide in *polynomial* time whether it possesses the Kakutani property, or not?

For the case that the algorithm has access to the closure system only via the corresponding closure operator, we give a *negative* answer to this question. More precisely, by constructing a Kakutani and a non-Kakutani closure system we show that all algorithms *distinguishing* between these two closure systems and, again, which have access to the closure systems via the closure operator only, require *exponentially* many closure operator calls in the worst case. The problem above can, however, be decided *efficiently* if we have some *a priori* information about the closure system at hand. In particular, for the case of d-ary[3] closure systems, the Kakutani property can be checked in $O\left(n^{cd}\right)$ time by a result of Chepoi (1994), where n is the size of the ground set and c is some constant.

Motivated by the negative result concerning problem (2) above for arbitrary closure systems, in our third research question we turn to the problem of *identifying* Kakutani closure systems. More precisely, we are interested in the following question:

(3) What are examples of *Kakutani closure systems* and how can we characterize them for specific domains?

We give an example of Kakutani closure systems over domains formed by the vertex sets of (undirected) graphs. When focusing on specific domains (e.g., vertex sets), we can further relax question (3) above by considering *sufficient* conditions of the Kakutani property only, instead of complete characterizations. Of course, incomplete characterizations may result in one-sided errors because they capture only particular fragments of Kakutani closure systems over the specific domain at hand. Still, it is an important research task to identify such fragments not only from a theoretical, but also from an application viewpoint. As we will show in the answer to question (5), such fragments allow for efficient computations of half-space separations. We note that several examples and characterizations of Kakutani closure systems have so far been published in the literature (see, e.g., Chepoi, 1994; Kubiś, 2002; van de Vel, 1984). Our example in the thesis is concerned with a *new* fragment of closure systems defined by the *geodesic closure* over graphs. It relies on a characterization of the Kakutani property in terms of the *Pasch Axiom* by Chepoi (1994). We generalize this result to *graph-structured partitioning* (GSP). That is, a graph-structured partitioning is a triple $\mathfrak{G} = (S, G, \mathcal{P})$, where S is a finite set, $G = (V, E)$ is a graph, and $\mathcal{P}$ is a partitioning of S into $|V|$ non-empty subsets. Furthermore, we use the connection to the Pasch Axiom to characterize a class of Kakutani closure systems over graphs in terms of $K_{2,3}$ as *forbidden minor*, where $K_{2,3}$ is the

[3]Each closed set can be represented by the union of closed sets generated by at most d elements.

complete bipartite graph with 2 resp. 3 vertices. As an immediate consequence of this result, we have that the family formed by the geodesic closed subsets of the vertex set of an *outerplanar* graph is always a Kakutani closure system.

All questions and problems considered so far and also regarded in the following sections are concerned with the *binary* case, which is in the focus of the thesis. That is, our goal is to separate *two* sets by two closed sets partitioning the domain (i.e., by two complementary half-spaces). Since half-space separations regard the two input sets (i.e., the positive and negative examples) *symmetrically*, it is a natural demand to generalize the story to multi-class separations, emphasizing that our focus in the thesis is on the binary case. Accordingly, our next question is somewhat orthogonal to all other questions and problems considered in the thesis:

(4) How can we generalize the Kakutani property from the binary case to *multi-class* separation problems?

In case of graphs, multi-class partitionings have already been investigated by Artigas et al. (2011), without considering the (generalized) Kakutani property. In contrast, we introduce the notion of the n-Kakutani property, which generalizes the Kakutani property from above to $n > 2$ input sets, and show the somewhat surprising result that the n_1-Kakutani property does not imply the n_2-Kakutani property in general, neither for $n_1 < n_2$ nor for $n_1 > n_2$.

1.1.2 Maximal Closed Set Separation in Finite Closure Systems

The focus of question (3) in the previous section was on relaxing the HSS problem by restricting it to special closure systems. We now turn to an approach relaxing the computationally intractable HSS problem in an entirely different way. In particular, our fifth question deals with the following problem:

(5) How to relax the HSS problem in a way that for *any* finite closure system and for *any* two sets with disjoint closures, one can *efficiently* compute two closed sets containing the input sets such that they always form a half-space separation if the underlying closure system is Kakutani; otherwise an "almost" half-space separation?

To arrive at such a relaxation, recall that the first problem in the previous section was to decide whether two sets are half-space separable, or not. In case of a negative answer, one may relax the problem by allowing for *non-perfect* solutions. That is, assuming that the closures of the input sets are disjoint, the combinatorial optimization task is to find two disjoint closed sets which contain the input sets and cover together *a maximum* number of elements of the domain. Again, it follows from our negative complexity result concerning question (1) that this problem is NP-hard. Indeed, knowing the optimal solution would immediately decide the HSS problem.

We therefore further relax the above optimization problem by giving up the demand of computing a separation of *maximum* cardinality. Instead, we consider the problem of returning some inclusion *maximal* solution only. That is, we are interested in returning two disjoint closed sets which contain the input sets and none of them is properly contained by a closed set satisfying the disjointness condition. This relaxed problem is referred to as the *maximal closed set separation* (MCSS) problem. As further contributions, we propose a simple generic *greedy* algorithm and prove that it

(i) solves the MCSS problem *correctly*,

(ii) is *optimal* concerning the number of closure operator calls, and

(iii) returns a *half-space separation* in case of Kakutani closure systems.

We prove (ii) by constructing a finite closure system and showing that all algorithms using strictly less closure operator calls than our greedy algorithm *fail* to return a correct solution. We note that the optimality of the greedy algorithm is *generic* in the sense that it holds for *all* finite closure systems. Regarding (iii) above, we prove an even stronger result. In particular, we show that our greedy algorithm provides an *algorithmic characterization* of Kakutani closure systems. That is, a finite closure system is Kakutani if and only if our greedy algorithm returns a half-space separation for *all* pairs of input sets with disjoint closures.

Note that the result above concerning the optimality of our *generic* greedy algorithm with respect to *all* finite closure systems does *not* imply that it cannot be improved for *specific* domains. This raises the following natural question:

(6) Is it possible to *improve* the number of closure operator calls needed by our generic greedy algorithm for closure systems over some *specific* domain?

We answer the above question *affirmatively*, by presenting an example. In particular, in case of *finite lattices* we show how to use *domain specific* knowledge so that the number of closure operator calls can be reduced drastically. More precisely, utilizing the compact representations of *ideals* and *filters* in lattices, we can *logarithmically* reduce the number of closure operator calls. We then give two examples of set separations in lattices, one concerning formal concept analysis (Ganter et al., 2005) and one inductive logic programming (Plotkin, 1970). These examples may be of some independent interest as well.

As mentioned above, our greedy algorithm provides an algorithmic characterization of Kakutani closure systems. Applying this generic result to lattices, as a further contribution we prove that our greedy algorithm provides an algorithmic characterization of *distributive lattices*. This result also demonstrates the *power* of our generic approach; it allows for a quite natural alternative proof of the result that distributive lattices are precisely those which satisfy the Kakutani property (van de Vel, 1984).

Our next question focuses on the practical usefulness of the theory developed above for *machine learning*. Motivated by the fact that linear separation by *maximum margin* hyperplanes (Boser et al., 1992) is typically more powerful in practice than that by *arbitrary* separating hyperplanes (Rosenblatt, 1958), it is natural to ask whether the *predictive performance* obtained by such separating half-spaces or by the maximal closed sets returned by the greedy algorithm can be improved by adapting the idea of linear separation with *maximum margin hyperplanes*. More precisely, we ask the following question:

(7) How to define the notion of *margins* for disjoint closed sets in case of finite closure systems and *how* to find maximum margin separations in the adapted setting?

To answer the first part of this question, we note that there are different options to define maximum margin hyperplanes in $\mathbb{R}^d$. In particular, Bennett and Campbell (2000) show that maximum margin hyperplanes can be defined in two equivalent ways: They are namely the hyperplanes maximizing the distance (i) to parallel supporting planes and (ii) to the disjoint convex hulls of the training sets. While from a mathematical programming perspective the first definition is preferred (c.f. Bennett and Campbell, 2000), the problem is that in case of finite closure systems, there is *no* notion of planes and also no concept of parallelism. We therefore use the second definition for our adaptation. It raises, however, a further problem: In general, for abstract closure systems we have no notion of a *metric*. We overcome this problem by using the more general concept of *monotone linkage functions* (Mullat, 1976), a generalization of metrics. More precisely, they are functions mapping each pair formed by a subset and an element of the ground set to some real number that are *anti-monotone* in their first parameter. While it holds that all metrics naturally give rise to monotone linkage functions, the converse of this implication is not true in general. Thus, if the underlying ground set is a metric space, then one can of course naturally define the semantically meaningful monotone linkage function. If it is not the case, then the monotone linkage function can be defined either by a domain expert or by directly using the underlying closure operator (see Section 2.3).

To answer the second part of question (7) above, we use a slightly modified version of our greedy algorithm to solve the MCSS problem. We prove that if the closures of the input sets are disjoint, then the modified version always returns a *maximum* (not only maximal) margin separation, regardless of the particular choice of the monotone linkage function, the closure system, and the input data.

We empirically evaluate our algorithms developed by using the results summarized above. That is, we are interested in the following question:

(8) Do maximum margin separations in finite closure systems result in an improved predictive performance, compared to arbitrary half-space and maximal closed set separations?

In particular, we answer the above question positively by considering two explicit domains: Finite point sets in $\mathbb{R}^d$ for different $d \in \mathbb{N}$ and vertices in graphs. On the one hand, we show that both algorithms, i.e., our MCSS algorithm and also the upgraded version using maximum margins, outperform the related baseline methods. On the other hand, we show that by using maximum margin separations, we can improve the results of our simple greedy algorithm.

1.1.3 Practical Aspects of Geodesic Closure Systems over Graphs

Finally, we discuss two practical aspects of finite closure systems by considering the specific case of *geodesic closure systems* over *graphs* (see, e.g., Pelayo, 2013). That is, the underlying ground set is the vertex set of a graph and a set of vertices is regarded to be *closed* (i.e., belongs to the closure system) if and only if for *all* vertex pairs in the set, it contains *all* vertices of *all* shortest paths between the pairs.

Approximating Geodesically Closed Sets in Graphs

In the theory developed above we have consistently used the closure operator as a black box function or oracle; its computational aspects have so far been swept under the carpet. Regarding the first practical aspect of geodesic convexity over graphs, we focus on the question of how to calculate geodesic closures over some (very) large graphs. It is a well-known result that for a graph G with n vertices and m edges, the geodesic closure of any subset of G's vertices can be calculated in time $O(nm)$. Clearly, this time complexity is *infeasible* in practice in case of *large* graphs. To make the algorithmic results developed in the thesis applicable to *large* graphs as well, in our next step we turn to the following question:

(9) Is it possible to calculate some tight *approximation* of the geodesic closure in *large* graphs in practically *feasible* time?

We give an *affirmative* answer to this question by proposing a heuristic consisting of three components. In particular, (i) we design an algorithm for sampling almost inclusion *maximal outerplanar spanning subgraphs* in time *linear* in the number of edges of the input graph and (ii) develop another algorithm for computing geodesic convex hulls in *outerplanar* graphs. Our algorithm is *faster* in practice than the corresponding standard algorithm because its complexity depends only on n (the number of vertices of the input graph) and on the *face number* of the outerplanar graph, which is some small number typically. In particular, it is *independent* of the size of the input set. Finally, (iii) the closure of a set of vertices in the input graph is then determined by its closures in the outerplanar samples. More precisely, a vertex is regarded as an element of the closure if it is contained in the closures of at most k of the outerplanar graphs, where k is some user-specified threshold. As two byproducts of this heuristic, the algorithm developed for (i) generating (almost) maximal outerplanar subgraphs in *linear time* and that designed for (ii) computing closure in outerplanar graphs in time in *typically linear time* may be of some independent interest.

To *experimentally* evaluate our heuristic on large graphs, we consider the task of approximating *core-periphery decomposition* of *large real-world* networks. This kind of decomposition, introduced recently in network mining by Marc and Šubelj (2018), is based on the notion of geodesic closures over graphs. They have observed that several real-world networks can be partitioned into a geodesically closed induced subgraph, the *core*, and its complement, the *periphery*. A somewhat surprising typical property of cores is that they are contained by the closure of (almost) all small sets of nodes, selected randomly. In fact, the intersection of typically about *three* closed sets, each generated by around *ten* nodes selected independently at random, becomes *fixed*, i.e., further closed sets generated in the same way all contain this intersection with high probability. We call this stable intersection the *geodesic core* of a network.

Our experiment clearly demonstrates that our heuristic is able to calculate a good approximation in a relatively short time, compared to the exact algorithm. For example, it was able to return close approximations in *five hours* or less for real-world networks with more than *20 million* edges. In contrast, the standard algorithm was *unable* to terminate within *50 days*.

Approximation of the Graph Tukey Depth

Finally, we present an application of the MCSS algorithm to approximate the *Tukey depth* of the vertices of a graph, a new notion introduced recently by Cerdeira and Silva (2021). The original notion of Tukey depth, introduced by Tukey (1975), is defined for finite point sets in $\mathbb{R}^d$ as follows: Given some *finite* subset S of $\mathbb{R}^d$, the Tukey depth of an element e of S is the cardinality of a smallest subset $S' \subseteq S$ containing e such that S' is contained by some half-space of $\mathbb{R}^d$. Or equivalently, it is $|S| - |S''|$, where $S'' \subset S$ is the largest set that is contained by a half-space and $e \notin S''$. This notion is used as an alternative *centrality measure* and has different applications in machine learning (Gilad-Bachrach et al., 2004) as well as in other fields (Dai et al., 2022; Mozharovskyi, 2015).

To adapt the above notion to finite closure systems, we make use of the facts that its definition is based on half-spaces and does not rely on some distance function. The adaptation is, however, more complex. The problem is that there is no guarantee that the underlying (abstract) closure system contains non-trivial half-spaces at all (see, again, the example in Figure 1.2(c)). In a recent work by Cerdeira and Silva (2021), the authors propose the following adaptation of ordinary Tukey depth in $\mathbb{R}^d$ to geodesic closure systems over graphs: Given a graph with vertex set V, the Tukey depth of a vertex v is $|V| - c$, where c is the cardinality of a largest closed set *not* containing v. In other words, for all subsets S of V of cardinality greater than c, the (geodesic) closure of S contains v. It follows from the definition that the Tukey depth of v can be regarded as another measure of *node importance* in networks (see, e.g., Newman, 2018, for further centrality measures). In particular, a *higher* Tukey depth of a node implies that the node is present in the convex hull of *all* "small" subsets of V. Although our focus is on geodesic closure systems over graphs, we note that the above adaptation of Tukey depth from $\mathbb{R}^d$ naturally applies to *all* finite closure systems as well. Furthermore, notice that if the geodesic closure system at hand is Kakutani then the maximum closed set not containing v in the definition above is at the same time a maximum half-space required by definition.

A computational *drawback* of the notion of graph (and also ordinary) Tukey depth is that computing the Tukey depth of a vertex is NP-hard (Cerdeira and Silva, 2021; Johnson and Preparata, 1978) in general. According to our results discussed above, while *maximum* closed set separation in finite closure systems is NP-hard, its relaxation, the *maximal* closed set separation problem can be solved in *polynomial* time (assuming for both problems that the underlying closure operator can be calculated in polynomial time). Thus, it is natural to ask whether graph Tukey depths can *effectively* be approximated by *maximal* separating closed sets. More precisely, we are interested in the following *practical* question:

(10) Is it possible to use the MCSS algorithm for computing close approximations of the graph Tukey depth?

We will answer this question positively by recalling that given some input sets with disjoint closures, the MCSS algorithm generates two *maximal* disjoint closed sets. Considering such a fixed separation, it is clear that the cardinality of each of the maximal closed sets bounds the Tukey depth of all elements not contained in this set. We assume that the repeated generation of *different* maximal disjoint closed sets[4] will improve the lower bound of the Tukey depth of the vertices more and more and hence the estimation error decreases. In order to *empirically* evaluate the approximation quality of our heuristic, we consider small graphs only. The reason is that for *large* graphs, we were unable to calculate the *exact* graph Tukey depth of the vertices. Our empirical results on small graph datasets clearly demonstrate that a very close approximation of the graph Tukey depth can be obtained by our heuristic based on maximal closed set separation.

1.2 Outline

The rest of the thesis is organized as follows. We recall the basic notions and notations in Chapter 2. In order to place the work into the scientific context, we collect the most important related work in Chapter 3. In the main part of the thesis, we answer the questions raised above. In particular, Chapter 4 is concerned with questions (1)–(4), Chapter 5 with questions (5)–(8), and Chapter 6 with questions (9) and (10). More precisely, in Chapter 4 we present the HSS problem and discuss Kakutani closure systems in general and for specific domains. Chapter 5 is devoted to the MCSS problem, the MCSS algorithm, and its algorithmic analysis. Moreover, we discuss and evaluate the usage of maximum margin separations in finite closure systems. In Chapter 6 we consider two practical aspects: One is concerned with analyzing geodesic closures over graphs and the other one with approximating the graph Tukey depth using the MCSS algorithm. Finally, in Chapter 7 we conclude the thesis by summarizing our main results, pointing out open questions, and discussing some problems for future research.

1.3 Previously Published Work

The thesis relies on the following joint papers with Tamás Horváth and Stefan Wrobel.

[4]This can be achieved by choosing different input sets for the MCSS algorithm. For example in the case of graphs we will use all pairs of neighbors in the graph as input.

Seiffarth, F., Horváth, T., and Wrobel S. (2019) Maximal closed set and half-space separations in finite closure systems. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019*, volume 11906 of *Lecture Notes in Computer Science*, pages 21–37. Springer. https://doi.org/10.1007/978-3-030-46150-8_2 The long version has been *submitted* to the Theoretical Computer Science journal.

Seiffarth, F., Horváth, T., and Wrobel S. (2020) Maximum margin separations in finite closure systems. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020*, volume 12457 of *Lecture Notes in Computer Science*, pages 3–18. Springer. https://doi.org/10.1007/978-3-030-67658-2_1

Seiffarth, F., Horváth, T., and Wrobel S. (2022a) A simple heuristic for the graph Tukey depth problem with potential applications to graph mining. In *Proceedings of the LWDA 2022 Workshops: FGWM, KDML, FGWI-BIA, and FGIR*, CEUR Workshop Proceedings. https://ceur-ws.org/Vol-3341/KDML-LWDA_2022_CRC_705.pdf

Seiffarth, F., Horváth, T., and Wrobel S. (2022b) A fast heuristic for computing geodesic closures in large networks. In *Discovery Science - 25th International Conference, DS 2022, Montpellier, France, October 10-12, 2022, Proceedings*, volume 13601 of *Lecture Notes in Computer Science*, pages 476–490. Springer. https://doi.org/10.1007/978-3-031-18840-4_34

Preliminaries 2

In this chapter we collect all the necessary notions and fix the notation.

Outline We start by giving the basic notions in Section 2.1 including that of graphs (Section 2.1.1) and lattices (Section 2.1.2). In case of lattices, we mention two examples explicitly: formal concept lattices in formal concept analysis and subsumption lattices in inductive logic programming. Section 2.2 is devoted to the basics of abstract set and closure systems. In particular, we recall the definitions of set and closure systems (Section 2.2.1) and give examples of particular closure systems over finite point sets in $\mathbb{R}^d$, graphs, and lattices (Section 2.2.2). Moreover, we provide all necessary notions of separations in finite closure systems (Section 2.2.3). Section 2.3 is concerned with monotone linkage functions that are used to define maximum margin separations in finite closure systems. Regarding the practical aspects of geodesic closure systems over graphs, we recall the notion of geodesic core-periphery decompositions in Section 2.4 and that of graph Tukey depth in Section 2.5. The measures used in our empirical evaluation are presented in Section 2.6. Finally, in Section 2.7 we give a detailed description of all the datasets used in this thesis. In addition, we provide examples to motivate most of the non-trivial definitions.

2.1 Basics

As usual, by $\mathbb{R}$ we denote the set of all *real numbers* and by $\mathbb{N}$ the set of all *natural numbers*, (0 is not included). Uppercase letters like E, X, Y are used to denote sets, and calligraphic letters such as $\mathcal{C}$ families of sets. Lowercase letters like c, d, n, m indicate the size of specific quantities. The *power set* of some set X is denoted by $2^X := \{Y : Y \subseteq X\}$. The set of numbers from 0 to n is abbreviated by $[n] := \{0, 1, \dots, n\}$. For some fixed ground set E and some subset $X \subseteq E$, we denote the set complement of X in E by X^c.

2.1.1 Graphs

For basic notions in graph theory, we refer to some standard textbook (see, e.g., Diestel, 2012). A graph $G = (V, E)$ is a tuple of sets with V being the set of vertices and $E \subseteq V \times V$ the set of edges. Referring to the underlying graph, the set of nodes, respectively the set of edges are denoted by $V(G)$, respectively $E(G)$ and $|V(G)|$ and $|E(G)|$ by n and m, respectively. Note that we use E for abstract ground sets, as well as, for the edge set of a graph. Nevertheless, it will always be clear from the context if we are considering finite ground sets or the edge set of a graph.

Unless otherwise stated, by graphs we mean finite *undirected* and *unweighted* graphs without loops and parallel edges. The *neighbors* of a vertex $v \in V(G)$ are denoted by $\Gamma(v) := \{u : \{v, u\} \in E(G)\}$. Furthermore, if it is clear from the context that the graph is undirected, we sometimes write uv for an edge $\{u, v\} \in E(G)$ for $u, v \in V(G)$. A *path* P of length k between two vertices $v_0, v_k \in V(G)$ is a sequence of edges $(e_1, e_2, \dots, e_k)$ with $e_1, \dots, e_k \in E(G)$ such that there exists a sequence of vertices $(v_0, v_1, \dots, v_k)$ with $e_i = \{v_{i-1}, v_i\}$ and all edges and vertices are pairwise disjoint. We refer to these sequences as $V(P)$ and $E(P)$. A path P is called *induced* if and only if there is no edge $f \in E(G)$ with $f = \{v_i, v_j\}$ for $|i - j| > 1$. Let P be some path with $V(P) = (v_0, v_1, \dots, v_k)$ then $\text{end}(P) = \{v_0, v_k\}$ denotes the *endpoints* of the path. In case of trees the unique path connecting two nodes u, v of a tree is denoted by $\text{Path}(u, v)$. A *shortest path* P between two vertices u, v is a path with $\text{end}(P) = \{u, v\}$ that has a minimum edge sequence length. The *interval* $[u, v]$ denotes the set of all vertices lying on all shortest paths between some vertices $u, v \in V(G)$. The set of all shortest paths in a graph is denoted by $\mathcal{P}_{\text{sp}}$.

A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and $\{v_1, v_2\} \in E(H)$ implies that $v_1, v_2 \in V(H)$. A subgraph H is an *induced subgraph* if for all $v_1, v_2 \in V(H)$ with $\{v_1, v_2\} \in E(G)$ it follows that $\{v_1, v_2\} \in E(H)$. A graph H is a *minor* of a graph G if it can be obtained by G using the following three operations on G: (i) edge contractions, (ii) edge deletions, and (iii) deletions of isolated vertices.

A graph G is called *biconnected* if the graph obtained by removing an arbitrary vertex and all its adjacent edges from G is connected regardless of the chosen vertex. A *biconnected component* B of a graph G is an induced subgraph of G which is biconnected and maximal concerning this property.

The graph $K_{2,3}$ which we use later on is the bipartite graph with two, respectively, three vertices in the two partitions and each vertex in one partition is connected to all vertices in the other partition.

Outerplanar graphs

In Chapter 6 we pay special attention to the class of *outerplanar* graphs (see, e.g., Chartrand and Harary, 1967). A graph G is *outerplanar* if it can be embedded in $\mathbb{R}^2$ so that no two edges cross each other (except possibly in their endpoints) and there exists a point $P \in \mathbb{R}^2$ such that each vertex of G can be reached from P by a simple curve that does not cross any of the edges. Such an embedding defines a set of connected pieces of the plane, called *faces*. Since G is finite, all faces are bounded except for one, the *outer* face; note that all points of the outer face fulfill the property of point P in the definition above. The bounded faces are called *interior* faces. All biconnected components, also called *blocks* of an outerplanar graph consist of a unique Hamiltonian cycle and a possibly empty set of (non-crossing) diagonals.

The *face number* of a biconnected outerplanar graph is defined by the number of its

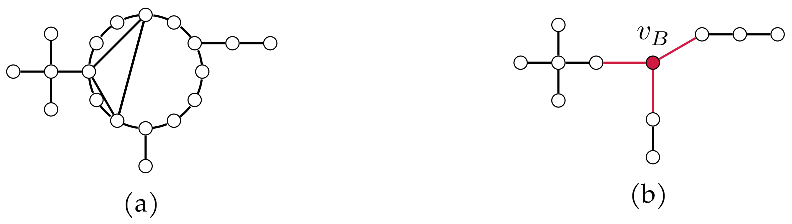


Figure 2.1: (a) Outerplanar graph with face number 4 and (b) its corresponding block and bridge tree. The vertex v_b (in red) is the newly created block vertex connected by the red edges to the rest of the block and bridge tree.

interior faces; the *face number* of an outerplanar graph G, denoted by $\Phi(G)$, is defined by the maximum of the face number of its biconnected components.

Edges not belonging to blocks are referred to as *bridges*. For an outerplanar graph G, let $\widetilde{G}$ denote the graph, called the *block and bridge tree* (BB-tree) of G, defined as follows (Horváth et al., 2010): For each block B of G, (i) introduce a new vertex, called *block vertex* v_B, (ii) remove all edges belonging to B, and (iii) for every vertex v of B, connect v with v_B by an edge if v is adjacent to a bridge or another biconnected component of G; otherwise remove v. It holds that $\widetilde{G}$ is a (free) tree and can be computed in $O(|V(G)|)$ time. Figure 2.1 shows an outerplanar graph (a) with face number 4 and its corresponding BB-tree in (b).

2.1.2 Lattices

For basic notions from *lattice theory*, the reader is referred to Davey and Priestley (2002) and Grätzer (2011). Let $(S; \leq)$ be a partially ordered set (or poset) and $X \subseteq S$. An element of S is the *supremum* (resp. *infimum*) of X, denoted $\sup X$ (resp. $\inf X$), if

(i) $\sup X \geq x$ (resp. $\inf X \leq x$) for all $x \in X$ and

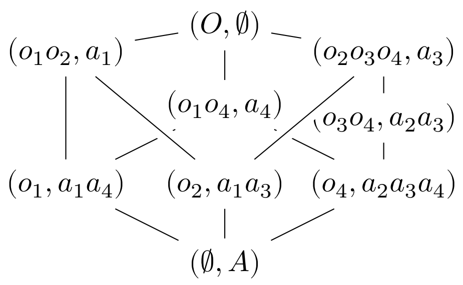
(ii) $y \geq \sup X$ (resp. $y \leq \inf X$) for all $y \in S$ with $y \geq x$ (resp. $y \leq x$) for all $x \in X$.

A poset $(L; \leq)$ is a *lattice* if $\sup X$ and $\inf X$ exist for all finite sets $X \subseteq L$. Thus, if L is finite then it is *bounded*, i.e., has a bottom and top element $\bot_L = \inf L$ and $\top_L = \sup L$, respectively. Unless otherwise stated, throughout this work by lattices we always mean *finite* lattices.

Let $(L; \leq)$ be a lattice. An element $x \in L$ is an *upper* (resp. *lower*) *cover* of $a \in L$ if $x > a$ (resp. $x < a$) and there is no $y \in L$ such that $x > y > a$ (resp. $x < y < a$). The set of upper (resp. lower) covers of a is denoted by $\mathrm{C}_{\uparrow}(a)$ (resp. $\mathrm{C}_{\downarrow}(a)$). A lattice L is called *distributive* if and only if $\inf\{a, \sup\{b, c\}\} = \sup\{\inf\{a, b\}, \inf\{a, c\}\}$ holds for all $a, b, c \in L$. A *sublattice* of L is a non-empty subset of L which is a lattice. An *ideal* I of L is a non-empty subset of L satisfying (i) $\sup\{a, b\} \in I$ for all $a, b \in I$ and (ii) $a \in I$ whenever $a \in L$, $b \in I$, and $a \leq b$. An ideal $I \subsetneq L$ is *prime* if $a \in I$ or $b \in I$ whenever $\inf\{a, b\} \in I$. The dual notions of ideals and prime ideals are called *filters* and *prime*

	a_1	a_2	a_3	a_4
o_1	1	0	0	1
o_2	1	0	1	0
o_3	0	1	1	0
o_4	0	1	1	1

(a) Formal context represented by a binary matrix M over $O \times A$.



(b) Concept lattice defined by M.

Figure 2.2: Example of a formal context (a) and its corresponding concept lattice (b).

filters, respectively. One can easily check that all ideals and filters of L are sublattices of L. Furthermore, as $|L| < \infty$ by assumption, an ideal I (resp. filter F) can be represented by sup I (resp. inf F). The ideal (resp. filter) of L with top (resp. bottom) element a is denoted by $(a]$ (resp. $[a)$). It follows from the definitions that the complement of a prime ideal of L is a prime filter of L and vice versa.

Formal Concept Lattice and Subsumption Lattice in Inductive Logic Programming

To prepare our examples on *formal concept analysis* and *inductive logic programming* in Section 5.2.3 we present two particular types of lattices.

Formal Concept Lattice A *formal context* is represented by a binary matrix M that is determined by the set O (objects) defining the rows of M and the set A (attributes) defining the columns of M. The entries of the matrix determine whether some object has (entry 1) some attribute, or not (entry 0). We already mentioned that the *lower* and *upper* adjoint functions defined on the binary matrix together form a Galois connection. In this particular case, the *lower adjoint* denoted by $'$ maps a subset $A_1 \subseteq A$ of the attributes to a subset of the objects

$$A_1' := \{o \in O : (o, a) = 1 \text{ for all } a \in A_1\} \subseteq O \enspace .$$

The *upper adjoint* also denoted by $'$ maps a subset $O_1 \subseteq O$ of the objects to a subset of the attributes

$$O_1' := \{a \in A : (o, a) = 1 \text{ for all } o \in O_1\} \subseteq A \enspace .$$

A pair (O_1, A_1) with $O_1 \subseteq O$ and $A_1 \subseteq A$ is called a *formal concept* if and only if $A_1' = O_1$ and $O_1' = A_1$. It is a well-known fact (see, e.g., Ganter et al., 2005) that the set of all concepts defined by M together with $(\emptyset, A)$ and $(O, \emptyset)$ form a lattice $(L; \leq)$ with the partial order "$\leq$" defined by

$$(O_1, A_1) \leq (O_2, A_2) \iff O_1 \subseteq O_2 \enspace .$$

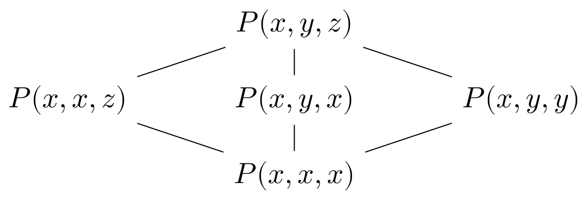


Figure 2.3: Subsumption lattice of the vocabulary consisting of a single predicate symbol and three variables x, y, z

This lattice is called *formal concept lattice*. Figure 2.2 shows an example of a formal context together with the corresponding formal concept lattice consisting of four objects and four attributes.

Subsumption Lattice A subsumption lattice (Khardon and Arias, 2006, see, e.g.,) is built by a partial order defined for first-order logic terms. For further details on *first-order logic* and *inductive logic programming* we refer to the book of Nienhuys-Cheng and de Wolf (1997). For simplicity we restrict the vocabulary to a single predicate symbol P of arity n and a set V of variables. We say an atom $P(t_1, \dots, t_n)$ with $t_1, \dots, t_n \in V$ *generalizes* $P(t'_1, \dots, t'_n)$ with $t'_1, \dots, t'_n \in V$, denoted by $P(t_1, \dots, t_n) \geq P(t'_1, \dots, t'_n)$, if there exists a function $\sigma : V \to V$ such that $P(\sigma(t_1), \dots, \sigma(t_n)) = P(t'_1, \dots, t'_n)$. Two atoms A_1, A_2 are *equivalent* if $A_1 \leq A_2$ and $A_2 \leq A_1$. Let L be a maximal set of P-atoms, each of the above form, that contains no two equivalent atoms. Clearly, each element of L can be represented by any atom from its equivalence class. The set L together with the partial order $\leq$ forms a lattice called *subsumption lattice*. Our restricted case holds that $(L; \leq)$ is a finite lattice. Furthermore, the top (resp. bottom) element of L is a P-atom such that all variables are pairwise different (resp. are the same). Figure 2.3 provides an example of a subsumption lattice of the vocabulary consisting of a single predicate symbol P and the three variables x, y and z.

2.2 Set and Closure Systems

In this section, we collect the necessary notions of *set systems*, *closure systems*, and *separations* in closure systems. For more details, we refer to the works of Chepoi (1994), Davey and Priestley (2002), and van de Vel (1993).

2.2.1 Definitions

A *set system* over a ground set E is a pair $(E, \mathcal{C})$ with $\mathcal{C} \subseteq 2^E$. Unless otherwise stated, all set systems considered in this paper are defined over *finite* ground sets. A (finite) set system $(E, \mathcal{C})$ is a *closure system* if it fulfills the following properties:

(i) $E \in \mathcal{C}$ and

(ii) $X \cap Y \in \mathcal{C}$ for all $X, Y \in \mathcal{C}$.

In addition to (i) and (ii), we assume without loss of generality that

(iii) $\emptyset \in \mathcal{C}$.[1]

It is a well-known fact (see, e.g., Davey and Priestley, 2002) that closure systems give rise to closure operators and vice versa. More precisely, a *closure operator* over E is a function $\rho : 2^E \to 2^E$ satisfying

i) $X \subseteq \rho(X)$, (*extensivity*)

ii) $\rho(X) \subseteq \rho(Y)$ whenever $X \subseteq Y$, (*monotonicity*)

iii) $\rho(\rho(X)) = \rho(X)$ (*idempotency*)

for all $X, Y \subseteq E$.

The connection between closure systems and closure operators is obtained by the following results (cf. Davey and Priestley, 2002). Proposition 2.2.1 shows, how to define the closure operator for a given closure system and Proposition 2.2.2, how to define the closure system for a given closure operator.

Proposition 2.2.1. *Let $(E, \mathcal{C})$ be a closure system. Then the map $\rho_{\mathcal{C}} : 2^E \to 2^E$ defined by*

$$\rho_{\mathcal{C}}(X) := \bigcap \{C \in \mathcal{C} : X \subseteq C\}$$

for $X \subseteq E$ is a closure operator.

Proposition 2.2.2. *Let ρ be a closure operator over E. Then $(E, \mathcal{C}_\rho)$ with*

$$\mathcal{C}_\rho = \{C \subseteq E : \rho(C) = C\}$$

is a closure system.

Depending on the context we sometimes omit the underlying closure operator from the notation and denote the closure system simply by $(E, \mathcal{C})$. The elements of $\mathcal{C}_\rho$ of a closure system $(E, \mathcal{C}_\rho)$ will be referred to as *closed* or *convex* sets. This latter terminology is justified by the fact that closed sets generalize several properties of *convex hulls* in $\mathbb{R}^d$.

Finally, we mention some particular type of closure systems that is defined by the following equivalence: A closure system $(E, \mathcal{C}_\rho)$ is called *d-ary* (Chepoi, 1994) if

$$C \in \mathcal{C}_\rho \iff \text{for all } X \subset C, |X| \leq n \text{ it holds } \rho(X) \subseteq C \ .$$

The definition implies that a closed set C of such a system can be written as the union of closed sets generated by at most d elements, i.e.,

$$C = \bigcup \{\rho(\{x_1, \dots, x_d\}) : x_1, \dots, x_d \in E\}^{2} \ .$$

[1] If $\emptyset \notin \mathcal{C}$ then (ii) implies that $Z = \bigcap_{X \in \mathcal{C}} X \neq \emptyset$. For the set system $(E \setminus Z, \{X \setminus Z : X \in \mathcal{C}\})$ we have that it is a closure system satisfying (iii) that has the same properties as $\mathcal{C}$.

[2] For example, geodesic closure systems over graphs, defined in the next section, are 2-ary closure systems because all closed sets are exactly the unions of the shortest paths between pairs of vertices in the set.

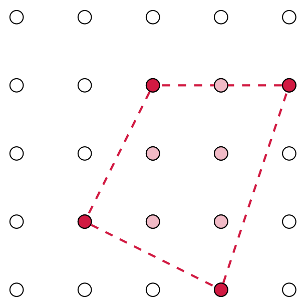


Figure 2.4: The α-closure of the four red points in a 5×5 grid in $\mathbb{R}^2$ is given by the intersection of the convex hull of the points (dashed lines) with the grid. The closure then consists of all the colored points (and *not* of the enclosed area).

2.2.2 Domain Specific Closure Systems

We use different domain-specific closure systems, e.g., over finite point sets in $\mathbb{R}^d$, graphs, and lattices and give the particular notions for the corresponding closure operators.

Finite convex hulls in $\mathbb{R}^d$

As a first example, we present the closure system defined by finite point sets in $\mathbb{R}^d$. Let E be a finite subset of $\mathbb{R}^d$ for some $d > 0$. Then the function $\alpha : 2^E \to 2^E$ defined by

$$\alpha(X) = \operatorname{conv}(X) \cap E \tag{2.1}$$

for all $X \subseteq E$ is a closure operator over E, where $\operatorname{conv}(\cdot)$ denotes the ordinary *convex hull* operator on $\mathbb{R}^d$. The corresponding closure system will be denoted by *α-closure system*. Figure 2.4 shows an example of this closure operator.

Geodesic Closures in Graphs

The second example is concerned with geodesic closures in graphs. The most familiar closure systems over graphs are those based on path sets. Such closure systems are usually called *interval* or *path closure systems* (Harary and Nieminen, 1981; Mulder, 1980). A detailed overview regarding geodesic closures in graphs can be found in the book of Pelayo (2013).

Proposition 2.2.3. *Let $G = (V, E)$ be a graph and $\mathcal{P}_{\text{sp}}$ the set of all shortest paths in G. Then the set system $(V, \mathcal{C}_\gamma)$ with*

$$\mathcal{C}_\gamma := \{C \subseteq V : V(P) \subseteq C \text{ for all } P \in \mathcal{P}_{\text{sp}} \text{ with } \operatorname{end}(P) \subseteq C\} \tag{2.2}$$

is a closure system.

Proposition 2.2.3 implies that a vertex set $X \subseteq V(G)$ is closed if and only if it contains *all* nodes from *all* the shortest paths for which the endpoints lie in X. The above defined

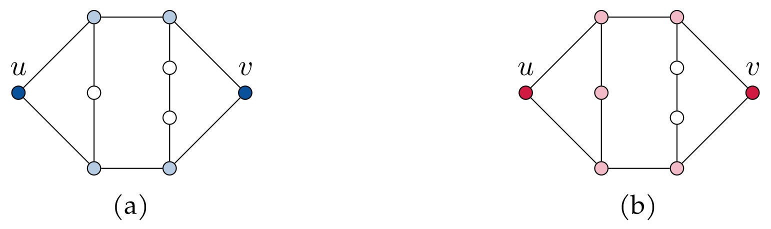


Figure 2.5: (a) The interval $[u, v]$ in blue and (b) the geodesic closure $\gamma(\{u, v\})$ in red.

closure system is usually called *geodesic* closure system. The corresponding closure operator γ is called *geodesic closure operator*. In Figure 2.5 we give an example of the above defined closure system by illustrating the interval $[u, v]$ in (a) and the geodesic closure $\gamma(\{u, v\})$ in (b). The algorithmic aspects, i.e., the task to compute the geodesic closure will be discussed in the next section.

In this thesis, our primary focus lies on the special case that the closure system in the underlying graph is based on the set of all shortest paths. In fact, the above also holds if the set of *all* shortest paths $\mathcal{P}_{\text{sp}}$ is replaced by some *arbitrary* subset of the set of all paths in the graph. For example, if considering the set of all induced paths (instead of shortest paths) the corresponding closure operator is called *monophonic* closure operator (Pelayo, 2013, see, e.g.,). Unless otherwise stated, in the rest of this thesis we always consider the geodesic closure system respectively operator.

Algorithmic Properties of the Geodesic Closure

For some graph $G = (V, E)$ the *geodesic closure* $\gamma(X)$ of $X \subseteq V(G)$ can be computed by iterating over *all* elements $u \in \gamma(X)$, starting with an arbitrary element of X, as follows: Let $X' \supseteq X$ be the set of elements in $\gamma(X)$ that have already been generated before we process the next element u. Then add $Y = \bigcup_{v \in X'} [u, v]$ to X', where Y can be calculated by solving the *single-source shortest path* (SSSP) problem (for unweighted graphs) from u to all elements of X'. After all elements in X' have been processed, we have $X' = \gamma(X)$. It is a folklore result that the SSSP problem can be solved with *breadth-first search* (BFS) in $O(n + m)$ time, where $n = |V(G)|$ and $m = |E(G)|$. Since $|\gamma(X)| = O(n)$ and $m = O(n^2)$, $\gamma(X)$ can be computed in *cubic* (i.e., $O(n^3)$) time (see, e.g., Pelayo, 2013).

In contrast, for outerplanar graphs, it suffices to consider only the pairs over X. More precisely, we will utilize the following result by Allgeier (2009):

Theorem 2.2.1. *Let G be an outerplanar graph. Then for all $X \subseteq V(G)$, $\gamma(X) = \bigcup_{u,v \in X} [u, v]$.*

Thus, in case of outerplanar graphs, it suffices to perform a breadth-first search only from the elements of X, resulting in the following corollary, by noting that $m = O(n)$ in case of outerplanar graphs:

Corollary 2.2.1. *Let G be an outerplanar graph and $X \subseteq V(G)$. Then $\gamma(X)$ can be computed in time $O(m|X|) = O(n|X|)$.*

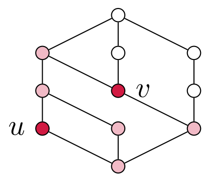


Figure 2.6: The colored elements in the lattice denote the λ-closure of the elements u, v.

Closed Sets in Lattices

Our third example is concerned with closure systems over finite lattices. Let $(L; \leq)$ be a finite lattice. Then the function $\lambda : 2^L \to 2^L$ defined by

$$\lambda : L' \mapsto \{x \in L \mid \inf L' \leq x \leq \sup L'\} \tag{2.3}$$

for all $L' \subseteq L$ is a closure operator, where $\inf L'$ (resp. $\sup L'$) denotes the greatest lower bound or bottom (resp. least upper bound or top) element of L'. The above defined function is usually called λ-closure operator. Figure 2.6 gives an example of the λ-closed set (colored lattice elements) $\lambda(\{u, v\})$.

In Lemmas 2.2.1 and 2.2.2 below we formulate some basic properties of finite lattices and λ-closure systems. Though most of the claims follow from basic properties of lattices, we provide all proofs for the reader's convenience.

Lemma 2.2.1. *Let $(L; \leq)$ be a finite lattice and $A, B \subseteq L$. Then the following statements are equivalent:*

(*i*) $\inf B \nleq \sup A$,

(*ii*) $[\inf B) \cap (\sup A] = \emptyset$,

(*iii*) *there exist an ideal $I \subseteq L$ and a filter $F \subseteq L$ with $I \cap F = \emptyset$ such that $A \subseteq I \wedge B \subseteq F$.*

Proof. For (i) $\Longrightarrow$ (ii), suppose for contradiction that $[\inf B) \cap (\sup A] \neq \emptyset$. Then there is an $x \in L$ with $\inf B \leq x$ and $x \leq \sup A$, contradicting (i). The proof of (ii) $\Longrightarrow$ (iii) follows directly from the fact that $[\inf B)$ is a filter and $(\sup A]$ an ideal. Regarding (iii) $\Longrightarrow$ (i), it must be the case that $\inf B \nleq \sup A$, as otherwise $\inf F \leq \inf B \leq \sup A \leq \sup I$, contradicting the disjointness of I and F. $\square$

Lemma 2.2.2. *Let $(L, \mathcal{C}_\lambda)$ be the λ-closure system over a finite lattice $(L; \leq)$ and $A, B \subseteq L$. Then $\lambda(A) \cap \lambda(B) = \emptyset$ if and only if there exist an ideal $I \subseteq L$ and a filter $F \subseteq L$ with $I \cap F = \emptyset$ such that $(A \subseteq I \wedge B \subseteq F)$ or $(B \subseteq I \wedge A \subseteq F)$.*

Proof. The proof of the "*if*" direction is immediate by $I, F \in \mathcal{C}_\lambda$. Regarding the other direction, we first claim that $\lambda(A) \cap \lambda(B) = \emptyset$ implies $\inf B \nleq \sup A$ or $\inf A \nleq \sup B$. Suppose for contradiction that $\inf B \leq \sup A$ and $\inf A \leq \sup B$. Then it follows $\inf B \leq \sup\{\inf A, \inf B\} \leq \sup B$ and $\inf A \leq \sup\{\inf A, \inf B\} \leq \sup A$, implying $\sup\{\inf A, \inf B\} \in \lambda(A) \cap \lambda(B)$, which contradicts $\lambda(A) \cap \lambda(B) = \emptyset$. The claim then follows from Lemma 2.2.1 by the symmetry of A and B. $\square$

2.2.3 Separations in Finite Closure Systems

The primary focus of this thesis is on half-space separations in finite closure systems. To formulate the separation problems in Chapters 4 and 5, we recall the general definitions concerning separations in finite closure systems, and turn to the generalization of linear separation in $\mathbb{R}^d$ by hyperplanes to that in *abstract* closure systems.[3] In the context of *machine learning*, one of the most relevant and natural questions concerning closure systems $(E, \mathcal{C})$ is whether two subsets of E are separable in $\mathcal{C}$, or not. We follow the generalization of half-spaces in Euclidean spaces introduced by Jamison (1974) to state the formal problem definition. More precisely, let $(E, \mathcal{C})$ be a closure system. Then $H \subseteq E$ is called a *half-space* in $\mathcal{C}$ if both H and its complement, denoted H^c, are closed (i.e., $H, H^c \in \mathcal{C}$).

More precisely, let $(E, \mathcal{C})$ be a closure system and $A, B \subseteq E$. Then A and B are

(i) *closed set separable* in $(E, \mathcal{C})$ if there exist $C_A, C_B \in \mathcal{C}$ such that $C_A \cap C_B = \emptyset$ and $A \subseteq C_A, B \subseteq C_B$,

(ii) *half-space separable* if there exist $H, H^c \in \mathcal{C}$ such that $A \subseteq H$ and $B \subseteq H^c$.

H and H^c together form a *half-space separation* of A and B. The following property will be used many times in what follows:

Proposition 2.2.4. *Let $(E, \mathcal{C}_\rho)$ be a closure system, $H, H^c \in \mathcal{C}$, and $A, B \subseteq E$. Then H and H^c form a half-space separation of A and B if and only if they form a half-space separation of $\rho(A)$ and $\rho(B)$.*

Proof. The "if" direction is immediate by the extensivity of ρ. The "only-if" direction follows from the fact that for any $S \subseteq E$ and $C \in \mathcal{C}_\rho$ with $S \subseteq C$ we have $\rho(S) \subseteq \rho(C) = C$ by the monotonicity and idempotency of ρ. $\square$

In line with the topological separation axioms, there also exist separation axioms for closure systems. We state the following four axioms introduced by Jamison (1974):

(S1) All singletons are closed (i.e., $\{e\} \in \mathcal{C}$ for all $e \in E$).

(S2) Two distinct elements can be separated by half-spaces (i.e., for $e, f \in E$ with $e \neq f$ there is some $H \in \mathcal{C}$ with $H^c \in \mathcal{C}$ and $e \in H, f \in H^c$).

(S3) Every closed set can be separated from a singleton not contained in the closed set by half-spaces (i.e., for some $e \in E, C \in \mathcal{C}$ with $e \notin C$ there is some $H \in \mathcal{C}$ with $H^c \in \mathcal{C}$, $e \in H$, and $C \subseteq H^c$).[4]

[3] For a detailed introduction into this topic see, e.g., the book of van de Vel (1993).

[4] Axiom (S3) is equivalent to the following axiom (cf. van de Vel, 1984): Every closed set can be obtained by the intersection of a family of half-spaces, i.e., for each $C \in \mathcal{C}$ there is a family $\mathcal{H}$ of half-spaces such that $C = \bigcap_{H \in \mathcal{H}} H$.

(S4) Two disjoint closed sets can be separated by half-spaces (i.e., for $C_1 \in \mathcal{C}, C_2 \in \mathcal{C}$ and $C_1 \cap C_2 = \emptyset$ there is some $H \in \mathcal{C}$ with $H^c \in \mathcal{C}$ such that $e \in H, C \subseteq H^c$).[5]

Note that the implications (S4) $\implies$ (S3) $\implies$ (S2) hold whenever (S1) holds. In general, we do not assume that singletons are closed, but in case of geodesic closures in graphs and closures in lattices the property (S1) holds true. An example where (S1) does *not* hold is the closure system induced by the Galois connection in case of formal concept analysis (see Section 2.1.2).

Our focus in this work is on closure systems satisfying the most restrictive axiom (S4). It is inspired by the Euclidean space counterpart result of Kakutani (1937) that disjoint convex sets can be separated by hyperplanes. This is motivated by the fact that basic machine learning algorithms, such as, for example, the Perceptron algorithm (Rosenblatt, 1958) or Support Vector Machines Boser et al. (1992) heavily rely on Kakutani's separation property in $\mathbb{R}^d$. Accordingly, one of our main goals is to study *Kakutani closure systems*, i.e., which can be characterized by (S4).

Notice that half-space separability in abstract closure systems does not preserve all natural properties of that in $\mathbb{R}^d$. For example, for any two *finite* subsets of $\mathbb{R}^d$ it always holds that they are half-space separable if and only if their convex hulls are disjoint. In contrast, Figure 1.2(c) shows that this equivalence does not hold for finite closure systems in general.

We will need the following weaker form of separation: Two closed sets $C_1, C_2 \in \mathcal{C}$ form a *maximal closed set separation* of A and B if and only if they form a closed set separation of A and B and there are no disjoint closed sets $C_1', C_2' \in \mathcal{C}$ with $C_1 \subseteq C_1'$ and $C_2 \subseteq C_2'$, where at least one of the containments is proper.

2.3 Monotone Linkage Functions

To adapt Vapnik's idea of maximum margin separation to (abstract) finite closure systems, we need some additional formal tools to quantify the closeness between subsets of the ground set. Such an abstract measure for the proximity between elements and subsets of a ground set is provided by *monotone linkage functions* introduced by Mullat (1976). These kind of functions preserve an important elementary property of distances from points to sets in metric spaces and can therefore be regarded as a very general "distance" concept. More precisely, a *monotone linkage function* over a set E is a map $l : 2^E \times E \to \mathbb{R}$ such that

$$X \subseteq Y \implies l(X, e) \geq l(Y, e)$$

holds for all $X, Y \subseteq E$ and $e \in E$. That is, l is *anti-monotone* with respect to set containment, which is an essential property satisfied by distances as well. Thus, all distances give rise to monotone linkage functions; the converse is, however, not true. Note that by applying monotone linkage functions to singletons in the first argument, we obtain

[5]This property which guarantees half-space separability for all disjoint closed sets will be referred to as Kakutani property in the following.

a pairwise proximity between the elements of the ground set. However, in contrast to metric spaces, the definition does not imply symmetry, i.e., $l(\{x\}, y)$ is not necessarily equal to $l(\{y\}, x)$. Furthermore, $l(X, e)$ is not required to be zero for $e \in X$.

Several examples of monotone linkage functions exist for finite and also infinite ground sets. Below we recall some of the most popular ones to illustrate the concept (see, e.g., Kempner et al., 1997, for further examples). The proof that the functions below are indeed monotone linkage functions is left to the reader.

(i) (*monotone linkage in* $\mathbb{R}^d$) For any distance D on $\mathbb{R}^d$, define

$$l : 2^{\mathbb{R}^d} \times \mathbb{R}^d \to \mathbb{R} \text{ by } l : (X, e) \mapsto \inf_{x \in X} \{D(x, e)\}$$

for all $X \subseteq \mathbb{R}^d$ and $e \in \mathbb{R}^d$.

(ii) (*monotone linkage in (weighted) graphs*) For a (weighted) graph $G = (V, E)$ define

$$l : 2^V \times V \to R \text{ by } l : (X, e) \mapsto \min_{x \in X} \{d(x, e)\}$$

for all $X \subseteq V$, where d denotes the (weighted) length of a (weighted) shortest path between vertices.

(iii) (*monotone linkage in graphs by maximum degree on induced subgraphs*) For a graph $G = (V, E)$, define

$$l : 2^V \times V \to \mathbb{R} \text{ by } l : (X, v) \mapsto \min_{x \in X} (\delta(v) - \delta_{G[X]}(x))$$

for all $X \subseteq V$ and $v \in V$, where $G[X]$ is the subgraph of G induced by X, $\delta(v)$ the degree of v in G, and $\delta_{G[X]}(x)$ the degree of x in $G[X]$.

(iv) (*monotone linkage between feature vectors*) Let $F \subseteq \mathbb{R}^d$ be a finite set consisting of d dimensional feature vectors. Then for every subset of feature vectors $X \subseteq F$ and a fixed feature vector $f \in F$ the linkage between f and X can be defined by

$$l(X, f) := \sum_{k=1}^{d} \min\{|f_k - x_k| : x \in X\}$$

where k denotes the k-th index of the d-dimensional vectors f respectively x.

Monotone linkage functions have been studied intensively in the context of *clustering* over set systems and convex geometries (Kempner and Levit, 2010; Kempner et al., 1997; Kempner and Muchnik, 2003). As mentioned above, we will use them for defining margins and maximum margin separations in *arbitrary* finite closure systems. For this purpose, we will use the following notion many times in what follows. A *monotone linkage closure system* (MLCS) is a triple $(E, \mathcal{C}_\rho, l)$ where $(E, \mathcal{C}_\rho)$ is a closure system and l is a monotone linkage function on E. Regarding the complexity results in Section 5.3.2 we assume that the linkage functions are given implicitly via an oracle. That is, for all $X \subseteq E$ and $e \in E$ the value of $l(X, e)$ is returned in *unit time* by an oracle.

So far, we have assumed that the monotone linkage function in the MLCS is given explicitly. It raises the legitimate question of why not use a metric instead of monotone linkage functions. Thus, we shortly discuss the case that the closure system does not give rise to a metric. We show that a monotone linkage function which is not necessarily a metric can always be defined directly from the structure of the closure system. We will call this special function the *natural monotone linkage function*.

(v) (*natural monotone linkage in finite closure systems*) Let $(E, \mathcal{C}_\rho)$ be some closure system. We denote by $\mathcal{G}(X, e) = \{Y \subseteq E, e \notin Y, e \in \rho(X \cup Y)\}$ the set of all $Y \subseteq E$ such that e does not lie in Y but in the closure of $X \cup Y$. Then the function $l : 2^E \times E \to \mathbb{R}$ defined by

$$(X, e) \mapsto \begin{cases} 0, & \text{if } e \in X \\ |E \setminus \rho(X)|, & \text{if } \mathcal{G}(X, e) \text{ is empty,} \\ min\{|Y| \in \mathcal{G}(X, e)\} & \text{o.w.} \end{cases}$$

is a monotone linkage function.

In particular, the linkage between X and e is the minimum number of elements distinct from e we need to add to X to generate e using the closure operator. Monotone linkage functions defined this way only depend on the closure system itself. This definition of monotone linkage functions is more of theoretical interest because in practice it is hard to determine the minimum generator sets and hence also the value of the monotone linkage function.

2.4 Geodesic Core-Periphery Decompositions

For an explicit application of mining in finite closure systems we look at geodesic closed sets in real-world graphs. More precisely, we are interested in approximating geodesic cores in large networks. Up to now, *geodesic cores* (Marc and Šubelj, 2018) are only *probabilistic* defined. Informally, the geodesic core of a graph consists of those nodes contained in every geodesic closed set for a small family of generator sets, each containing a few random nodes. Of course, the geodesic core defined in this way may be empty, but this is not the case for most social networks. Incorporating the results of Marc and Šubelj (2018), we define the geodesic core of a graph G by

$$\mathcal{C} := \bigcap_{j=1}^{i} C_j,$$

where i is the *smallest* integer satisfying $\bigcap_{j=1}^{i} C_j = \bigcap_{j=1}^{i+1} C_j$ and $C_j = \gamma(X_j)$ is the geodesic closure of $X_j \subseteq V(G)$ containing $k > 0$ nodes selected independently and uniformly at random from $V(G)$. The experiments in Section 6.1.3 with large real-world networks show that for $k \approx 10$, the core (if it exists) does *not* depend on the particular choice of the generator elements. The *core-periphery decomposition* of a graph is composed of the

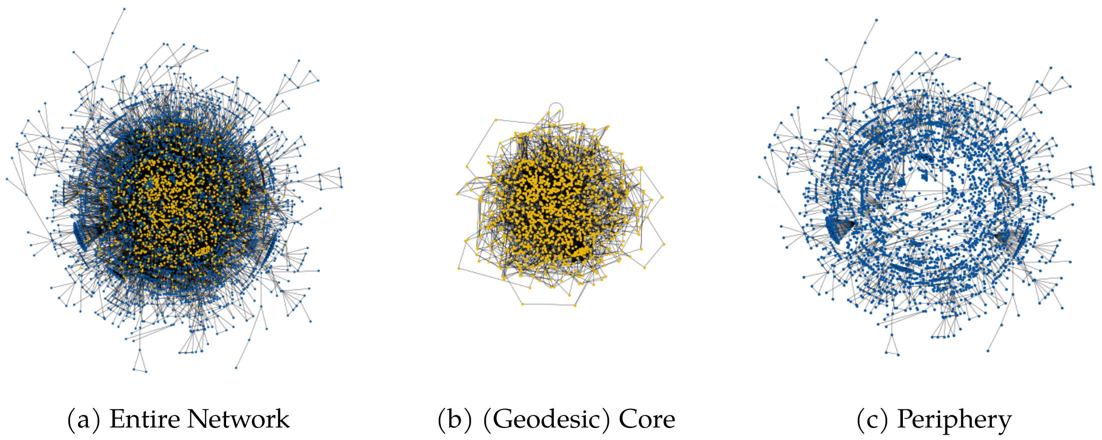


Figure 2.7: (a) CA-GrQc network Leskovec and Krevl (2014), (b) its (geodesic) core, (c) its periphery

subgraph induced by the core nodes and that by the remaining nodes, called *periphery*. In Figure 2.7 we give a visual example of the core-periphery decomposition of the CA-GrQc network (Leskovec and Krevl, 2014)[6].

2.5 Tukey Depth in Finite Closure Systems

Given the notion of half-spaces in finite closure systems it is possible to generalize the definition of Tukey Depth in $\mathbb{R}^d$ (Tukey, 1975) to finite closure systems. Each hyperplane in $\mathbb{R}^d$ separates the space into two disjoint half-spaces. For some finite point set $E \subset \mathbb{R}^d$ the Tukey depth of some element $e \in E$ is given by the minimum number of elements lying in the same half-space as e looking at all possible separating hyperplanes not containing some element from E. Johnson and Preparata (1978) have shown that it is NP-hard to compute the Tukey Depth in general. Since we show in Section 4.1 that it is not possible to find separating half-spaces in arbitrary closure systems we adapt the slightly weaker definition by Cerdeira and Silva (2021) proposed in case of geodesic closure systems to *arbitrary* finite closure systems $(E, \mathcal{C})$.

Definition 2.5.1. *Let $(E, \mathcal{C})$ be an arbitrary closure system. Then for an element $e \in E$ the Tukey depth is defined by*

$$\operatorname{td}(e) := |E| - \max_{C \in \mathcal{C}} \{|C| \;:\; \rho(\{e\}) \cap C = \emptyset\} \enspace .$$

The corresponding *Graph Tukey depths* problem is defined as follows: Given a graph G compute $\operatorname{td}(v)$ for all vertices $v \in V(G)$. In case of Kakutani closure systems we can derive the following result.

[6]This network is built by the co-authorships in the general relativity and quantum cosmology community.

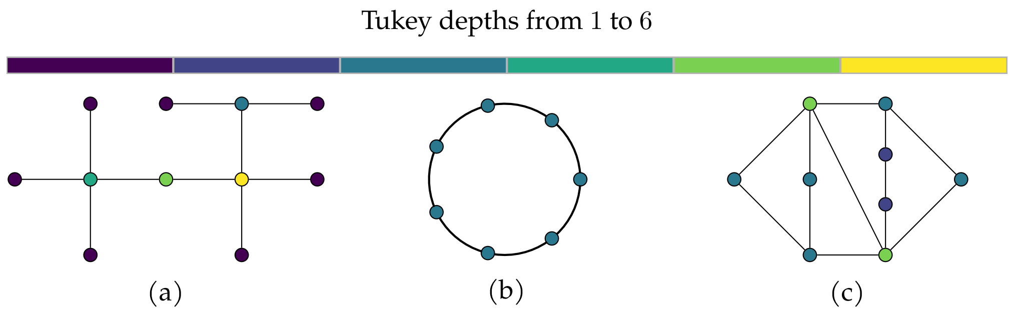


Figure 2.8: Vertex Tukey depths (from 1 to 6 denoted by the colors from above) for different types of graphs: (a) a tree, (b) a circle and (c) some planar graph

Proposition 2.5.1. *Let $(E, \mathcal{C})$ be a Kakutani closure system. Then for all $e \in E$ it holds*

$$\text{td}(e) = \min\{|H| \ : \ H, H^c \in \mathcal{C}, e \in H\} \ .$$

Proof. By definition of Tukey depth in finite closure systems it holds that $\text{td}(e) = |E| - |C^*|$ where C^* is a largest closed set that is disjoint to $\rho(\{e\})$, i.e., $C^* \in \arg\max_{C \in \mathcal{C}}\{|C| \ : \ \rho(\{e\}) \cap C = \emptyset\}$. Since by assumption the closure system is Kakutani we can separate the two disjoint closed sets $\rho(\{e\})$ and C^* by half-spaces. Let H, H^c such a half-space separation with $C^* \subseteq H$. It follows that $H = C^*$ as otherwise C^* is not a closed set of maximal cardinality that is disjoint to $\rho(\{e\})$. Since C^* is a largest half-space that is disjoint to $\rho(\{e\})$ the claim follows by noting that $\text{td}(e) = |(C^*)^c|$ is the size of a smallest half-space including $\rho(\{e\})$. □

Proposition 2.5.1 shows that for Kakutani closure systems the Tukey depth definition matches the original definition in $\mathbb{R}^d$ using half-spaces. In Figure 2.8 we give an example of the graph Tukey depths of vertices for different graphs for the geodesic closure.

2.6 Performance Measures

In Section 5.4 we analyze and compare the performance of different algorithms concerning their *accuracy* and *coverage*. To define the accuracy and coverage we first state the underlying task. Let $(E, \mathcal{C})$ be a finite closure system and $L_1, L_2 \subseteq E$ a complete labeling of the ground set, i.e., $L_1 \cap L_2 = \emptyset$ and $L_1 \cup L_2 = E$. The input to the algorithms are training sets $T_1 \subseteq L_1, T_2 \subseteq L_2$. The output sets of the algorithms are denoted by $C_1 \supseteq T_1, C_2 \supseteq T_2$. Moreover, we assume that $\rho(T_1) \cap \rho(T_2) = \emptyset$ and by definition of the algorithms it holds that $C_1 \cap C_2 = \emptyset$. By $T := T_1 \cup T_2$ we denote the set of all training samples. The sets of labels without training samples are denoted by $\widehat{L}_1 := L_1 \setminus T_1$ respectively $\widehat{L}_2 := L_2 \setminus T_2$. The output sets without training samples are denoted by $\widehat{C}_1 := C_1 \setminus T_1$ respectively $\widehat{C}_2 := C_2 \setminus T_2$ and the ground set E without all the training samples is denoted by $\widehat{E} := E \setminus T$. Given some finite closure system $(E, \mathcal{C})$, a complete labeling L_1, L_2 of the ground set and training samples T_1, T_2, we define the accuracy and coverage of the output C_1, C_2 as follows:

$$\mathrm{Acc}(C_1, C_2) := \frac{|(\widehat{C_1} \cap L_1) \cup (\widehat{C_2} \cap L_2)|}{|\widehat{C_1} \cup \widehat{C_2}|} = \frac{1}{|\widehat{C_1} \cup \widehat{C_2}|} \sum_{i=1}^{2} |\widehat{C_i} \cap L_i|,$$

and

$$\mathrm{Cov}(C_1, C_2) := \frac{|\widehat{C_1} \cup \widehat{C_2}|}{|\widehat{E}|} \quad .$$

We note that for the coverage it holds $\mathrm{Cov}(C_1, C_2) = 1$ if and only if our algorithm is able to find separating half-spaces.

In Section 6.1 we approximate the geodesic core of a graph and measure the overlap between the exact core and our approximation using the *Jaccard similarity*. This quality measure is defined as follows: Let $X, Y \subseteq E$ be some finite subsets of a ground set E (for example the set of the core nodes and its approximation). Then

$$\mathrm{Jacc}(X, Y) := \frac{|X \cap Y|}{|X \cup Y|} \quad .$$

denotes the *Jaccard similarity* between the sets X and Y. We note that $\mathrm{Jacc(X, Y)} = 1$ if and only if $X = Y$ and $\mathrm{Jacc(X, Y)} = 0$ if and only if $X \cap Y = \emptyset$.

2.7 Datasets

In this section, we give the information about the datasets used in this thesis. We use different datasets concerning finite point sets (see Table 2.1) and graphs to evaluate the maximal closed set separation (MCSS) algorithm in Section 5.4.1. Regarding the evaluation of our heuristic in Section 6.1 we use different synthetic and large real-world graphs (see Table 2.3). For the evaluation of our algorithm that approximates the graph Tukey depths in Section 6.2 we use different small graph datasets (see Table 2.2).

2.7.1 Finite Point Sets

The data presented in Table 2.1 and used in Section 5.4.1 consists of three self-created synthetic datasets Synthetic2D, Synthetic3D and Synthetic4D and the binary labeled datasets Banana, Banknote and DeltaAilerons from the UCI repository (Dua and Graff, 2017) and OpenML (Vanschoren et al., 2013). Banana is a synthetic dataset that consists of two banana shape classes, i.e., in particular, the classes are not separable by disjoint convex sets. Banknote consists of four-dimensional feature vectors; the features correspond to particular values of a wavelet transform representing real and fake banknotes images. DeltaAilerons is a binarized version of the Ailerons dataset; the attributes describe the status of an aircraft and the goal is to predict some control action.

The self-created synthetic datasets consist of two blobs, each with 500 points, sampled from two Gaussian distributions with different means but the same variance. We use

Name	Size	Dimen-sion	Majority Class	Training Set Size per Class
Synthetic2D	1000	2	0.5	1–50
Synthetic3D	1000	3	0.5	1–50
Synthetic4D	1000	4	0.5	1–50
Banana	5300	2	0.55	1–5
Banknote	1372	4	0.56	1–60
DeltaAilerons	7129	5	0.53	1–13

Table 2.1: Binary labeled synthetic and real-world datasets (Dua and Graff, 2017; Leskovec and Krevl, 2014) consisting of numerical feature vectors. The column *Size* denotes the size of the data, *Dimension* the dimension of the feature vectors, *Majority Class* the relative size of the larger class, and *Training Set Size per Class* denotes the range of the number of training samples used in our experiments in Section 5.4.1.

only such sampled data in which the two classes are separable by a hyperplane. As the name suggests the difference between the datasets is that the elements come from $\mathbb{R}^2$, $\mathbb{R}^3$ or $\mathbb{R}^4$. For the experiments we used 1000 variants of Synthetic2D, Synthetic3D, and Synthetic4D using the same Gaussian distributions but with differently drawn samples.

2.7.2 Graphs

This section is devoted to different synthetic and real-world graph datasets.

Small Graphs Datasets The small graphs dataset (Table 2.2) contains 19 real-world graph datasets (Morris et al., 2020) consisting of small graphs with less than 100 nodes on average. Disconnected graphs are removed from the original datasets. It is used to evaluate the graph Tukey depth approximation algorithm introduced in Section 6.2.

The following three datasets are used for the experiments in Section 6.1. The first two datasets are generated by sampling from the Erdős-Rényi model (Erdos et al., 1960). The third one contains several medium to large real-world graphs (Leskovec and Krevl, 2014).

Erdős-Rényi I This dataset contains *small* Erdős-Rényi random graphs used to evaluate Algorithm 4 generating outerplanar spanning subgraphs.[7] The size of the graphs varies, ranging from $n = 100$ to $n = 500$ with a step size of 100 and for ten different edge probabilities, ranging from $p = 0.05$ to $p = 0.14$ with a step size of 0.01. For each of the

[7]We use only small graphs because testing how many edges can be added to the graph without destroying outerplanarity is in $O(nm)$.

Data	Number of Graphs	Average Number of Nodes	Average Number of Edges
BZR	405	35.75	38.36
PTC_MM	336	13.97	14.32
COX2	467	41.22	43.45
Cuneiform	267	21.27	44.80
DHFR	756	42.43	44.54
PTC_FR	351	14.56	15.00
PTC_FM	349	14.11	14.48
MUTAG	188	17.93	19.79
PTC_MR	344	14.29	14.69
KKI	83	26.96	48.42
IMDB-BINARY	1000	19.77	96.53
NCI1	3530	29.27	31.88
Peking_1	85	39.31	77.35
MSRC_21C	209	40.28	96.60
MSRC_9	221	40.58	97.94
OHSU	79	82.01	199.66
ENZYMES	569	31.68	61.44
MSRC_21	563	77.52	198.32
COIL-DEL	3900	21.54	54.24

Table 2.2: Graph data of different sizes selected from Morris et al. (2020). Disconnected graphs are removed from the original datasets.

50 different configurations of (n, p), 100 *connected* Erdős-Rényi random graphs have been generated.

Erdős-Rényi II This dataset also contains Erdős-Rényi connected random graphs with 10 different sizes from $n = 1\,000$ to $n = 10\,000$ with a step size of $1\,000$ and with edge probabilities ranging from $p = 0.006$ to $p = 0.02$, with step size 0.002. Below $p = 0.006$, the graphs were too sparse for our purpose. For $n = 10\,000$ and $p = 0.02$, the graphs contain around 1,000,000 edges. For all of the 80 configurations of (n, p), 100 *connected* Erdős-Rényi random graphs have been generated.

Real-World Large Graphs This dataset, see Table 2.3, contains 15 real-world graphs (Leskovec and Krevl, 2014). In case of disconnected graphs, only their largest connected components were considered[8]. The dataset is used for the evaluation of the approximation of geodesic cores in Section 6.1.

[8]The numbers in Table 2.3 already denote the largest connected component.

Graph	#Vertices	#Edges	Density
com-Orkut	3,072,441	117,185,083	2.5e-05
soc-LiveJournal1	4,843,953	43,362,750	3.7e-06
soc-pokec-relationships	1,632,803	22,301,964	1.7e-05
com-youtube.ungraph	1,134,890	2,987,624	4.6e-06
com-dblp.ungraph	317,080	1,049,866	2.1e-05
com-amazon.ungraph	334,863	925,872	1.7e-05
Slashdot0902	82,168	582,533	1.7e-04
Cit-HepPh	34,401	420,828	7.1e-04
Cit-HepTh	27,400	352,059	9.4e-04
CA-AstroPh	17,903	197,031	1.2e-03
CA-CondMat	21,363	91,342	4.0e-04
CA-HepPh	11,204	117,649	1.9e-03
Wiki-Vote	7,066	100,736	4.0e-03
CA-HepTh	8,638	24,827	6.7e-04
CA-GrQc	4,158	13,428	1.6e-03

Table 2.3: Large real-world networks from Leskovec and Krevl (2014) with number of vertices (n), number of edges (m) and density. The networks are sorted by nm.

Related Work 3

This chapter overviews the most important results from the research fields related to the topic of this thesis. In particular, we discuss papers that have influenced our work, for example, by using and extending their results and definitions. In addition, we also mention the origins of the different research areas used throughout this thesis. More precisely, we discuss theoretical results concerning separations in closure systems from the field of *computational geometry* and papers adapting classical *machine learning* algorithms and paradigms, especially linear and maximum margin separations. We start by recalling the roots and the development of *abstract closure systems*. Like in many other fields that have developed over time, many different notions of similar or equivalent objects exist. We will shortly mention the various notions of closure systems used by different authors. Since we are especially interested in mining and learning over graphs, we also recall some specific results concerning *graph geodesic convexity*.

Our primary motivation to consider the adaptation of linear separation from $\mathbb{R}^d$ to finite closure systems is because of its simplicity and basic nature and, at the same time, its success in solving various real-world machine learning tasks in entirely different fields. Similarly to the evolution in $\mathbb{R}^d$ from arbitrary linear to maximum margin separations, we first look at arbitrary separations and then define maximum margin separations in finite closure systems. Accordingly, we overview the related techniques that generalize maximum margins and compare them with our method.

Finally, we recall the most important related work about mining and learning in finite closure systems and put them into the context of this thesis. One main difference between this thesis and the related papers is that they are all restricted to some fixed domain and do *not* regard linear separations in finite closure systems on a generic level. Since the origins of closure systems date back more than 100 years and the ones of classical machine learning more than 60 years, it is impossible to give a complete list of all related papers. In this chapter we therefore restrict the discussion to the most important results relevant to this thesis.

Outline The rest of this chapter is structured as follows. In Section 3.1 we discuss the most relevant work on closure systems. That includes the particular case of geodesic closure systems (Section 3.1.1) and results concentrating on separations in finite closure systems (Section 3.1.2). In Section 3.2 we recall the origins and basic concepts of linear separations in machine learning. In particular, we mention several approaches that introduce maximum margin separations for different domains (Section 3.2.1). Finally, in Section 3.3, we list the most relevant practical applications concerning mining and learning in finite closure systems.

3.1 Abstract Closure Systems

As mentioned in Chapter 1, the origins of abstract convexity theory go back to the early 19th century to the work of some famous mathematicians including Ernst Schröder, Georg Cantor, and Richard Dedekind. The first formal approaches go back to the works of Riesz (1909) and Moore (1910). While Riesz concentrates more on the topological aspects of convexity, Moore already defines some kind of closure operator. The consideration of abstract closure systems was mainly motivated by problems in different theoretical areas, including topology, algebra, logic, and computational geometry.

It is challenging to keep track of the literature because there are various definitions for closure systems, all describing the same or at least similar concepts. For example, Levi (1951) and also Kay and Womble (1971) speak of *convexity structures* or *convexity spaces*, while Jamison (1974) uses the notion of *alignments*. Alignments are somehow specially designed for infinite closure systems, including the additional assumption that unions of a nested family of totally ordered closed sets are also axiomatically defined as closed. Assuming the axiom of choice and using Zorn's Lemma, this extra condition implies that for each element from the ground set, there exists a maximal closed set that does not contain this element (Duchet, 1987). For our purposes, as we are considering *finite* closure systems only, this assumption is not necessary. Furthermore, there exist various concepts which consider special cases of closure systems or definitions that slightly differ from the version used in this thesis. Without the sake of completeness, we only mention the work by Bryant and Webster (1972, 1973, 1977). They study so-called *convexity systems*, which are based on closed intervals. In particular, according to their definition, "closed sets" are always unions of closed intervals (examples are convex hulls in $\mathbb{R}^d$ and geodesic convexity in graphs). The reason we mention their work is that they are among the first who regarded hyperplane separations in convexity systems (Bryant and Webster, 1973).

Abstract half-spaces, as used in this thesis, i.e., closed sets with closed complements, were introduced as *hemispaces* in the thesis of Jamison (1974). He also introduced four basic separation properties for closure systems (S1)-(S4) that we discuss in Section 3.1.2 in more detail. Finally, we mention two works that serve as good surveys. While the paper by Duchet (1987) provides an excellent summary of the development of different branches of convexity theory, the book of van de Vel (1993) is a detailed survey on closure systems.

Closure systems are widely used in different areas of computer science, in particular, in data mining and machine learning. Examples include the application of Galois connections to *itemset mining* (Pasquier et al., 1999) and *formal concept analysis* (Ganter et al., 2005) mentioned already in Chapter 1.

3.1.1 Convexity in Graphs

Since we have a special focus on the domain of graphs in this thesis, we briefly discuss the most important work on convexity in graphs. *Geodesic convexity* in graphs was first studied by Mulder (1980) and Harary and Nieminen (1981). We note that most of the literature is about geodesic convexity, which relies on the set of all shortest paths, with the remark that there are other graph convexities which rely on different sets of paths. For example, instead of considering the set of *all* shortest paths, Farber and Jamison (1987) consider only shortest paths up to a certain length. Another well-studied graph convexity is the *monophonic convexity* (Duchet, 1988), which relies on the set of all chordless, also known as induced paths. For an excellent overview of the most important results, the reader is referred to the book of Pelayo (2013).

Several interesting graph theoretic parameters can be expressed by using (geodesic) convexity in graphs. One main research direction is the adaptation of Caratheodory, Helly, and Radon theorems from $\mathbb{R}^d$ to graph convexity (Duchet and Meyniel, 1983). Another well-studied problem is to determine the *geodetic number* (Harary et al., 1993) or *hull number* of a graph (Everett and Seidman, 1985) and its characterization for different types of graph classes (see, e.g., Araujo et al., 2013; Dourado et al., 2010). The geodetic respectively hull number is the minimum size of a vertex set such that the closure of the set generates the whole graph. As a closely related work, we also mention the papers by Artigas et al. (2011, 2007) on geodesic convexity. Similarly to our approach, they look at covers and partitionings of graphs with convex sets. Their work allows us to prove the hardness of half-space separations in general closure systems. Regarding the practical applications of geodesic convexity to mining and learning in graphs, we discuss some recent results later in Section 3.3.

3.1.2 Theoretical Work on Separations

This section is devoted to the most important results about closed set separations in finite closure systems. Most of this results are purely theoretical, without any experiments or explicit applications. Nonetheless, they form the basis for all our applications in finite closure systems.

Like in many other fields, the research started with separations in $\mathbb{R}^d$. In particular, the origins of set and especially, convex set separations arose in the fields of *geometry*, and *topology*. One of the first and central results in this context is Farkas' Lemma (Farkas, 1902), which is motivated by separating disjoint convex sets via hyperplanes. Extensions are the well-known hyperplane separation theorem by Minkowski (1911), which states that disjoint convex sets are always separable by hyperplanes and the Hahn-Banach theorem (Banach, 1929; Hahn, 1927) in topology. Since we are considering *finite* closure systems, we are interested in the geometric version of this separation result attributed to Kakutani (1937) and used in the related literature (see, e.g., Chepoi, 1994; Kubiś, 2002; van de Vel, 1993). It states that disjoint convex sets in $\mathbb{R}^d$ can be separated by complementary convex sets, i.e., half-spaces. Similarly to Euclidean spaces, different kinds of separations can be considered for (finite) closure systems using the definition of closed sets. Jamison (1974) formulated four separation axioms, (S1) to (S4), that are similar to the topological separation axioms (see Section 2.2.3).

Since we are looking for partitionings of the space by disjoint closed sets, in this work, we concentrate on the most restrictive axiom (S4) of Jamison. This axiom is inspired by the Euclidean space counterpart result of Kakutani (1937) that convex sets can be separated by half-spaces. In other words, our goal is to analyze different closure systems and formulate characterization results that guarantee the Kakutani property (S4), which, in turn, allows for complete separation. The Kakutani property is essential in the case of *complete* binary classification via disjoint closed sets. Such half-space separations have been considered in different domains extending the separation results of $\mathbb{R}^d$ to other structures. The first results towards this direction are due to Stone (1938) and Tukey (1942). In particular, while Stone (1938) considers the connection between distributive lattices and lattice partitionings into prime filters and ideals, Tukey (1942) considers half-space separations of real linear spaces. These characterization results of the (S4) property are then generalized by the work of Ellis (1952). while the results of Bair (1975) are restricted to straight line spaces, the convexity spaces defined by Bryant and Webster (1973) to the case of ordinary closure systems generated by the union of closed intervals. One of the most detailed works concerning the (S4) property is due to Chepoi (1994). He gives several characterizations of (S4) closure systems for n-ary convexities, including interval convexities (e.g., graph convexities and lattices). One of his main results we use throughout the thesis is the characterization of Kakutani closure systems via the *Pasch Axiom*. Further results about separations in closure systems are provided by (Kubiś, 2002; van de Vel, 1982, 1984).

3.2 Classical Machine Learning

The development of machine learning algorithms has proceeded in several stages. In 1958, Rosenblatt (1958) published his seminal paper about the Perceptron algorithm. This first stage ended with the publication of Minsky and Papert (1987)[1] in 1969, showing that the XOR-problem is not solvable by the Perceptron algorithm. The topic did not receive much attention until the work of Hopfield (1982) about Hopfield networks and the usage of back-propagation by Rumelhart et al. (1986). These developments have made it possible to "learn" more complex networks. Thus, the increasing computing power enabled the solution of more complex and attractive learning problems. Examples are multi-layer neural networks with non-linear activation functions that can solve the XOR problem and are universal approximators (Hornik et al., 1989). Although this is a fascinating field of research, we are more interested in the fundamental properties of learning, more precisely, those of linear separations in $\mathbb{R}^d$. Moreover, we are also interested in the main idea of support vector machines that extend the Perceptron algorithm by integrating margins. Ultimately, we are looking at this extension because its concept, as we show in this thesis, can be transferred to finite abstract closure systems.

[1]This references a newer version of the original book from 1969.

While the origins of the theory of support vector machines (SVMs) go back to the work of Vapnik and Chervonenkis (1974), the support vector machine algorithm is presented for the first time in the paper by Boser et al. (1992), including the usage of kernel functions. Cortes and Vapnik (1995) improved the algorithm by considering non-separable data and Freund and Schapire (1999) proposed a simpler version that uses less computation time. Bennett and Campbell (2000) analyzed the backgrounds of the SVM method and mentioned three main aspects, namely, "margins", "duality" and "kernels". While duality and kernels rely on the fact that the underlying space is an inner product space, the definition of margin only needs some similarity measure. Moreover, they compare two different methods of finding the maximum margin separating hyperplane. One of them computes the hyperplane by "maximizing the margin between parallel supporting planes"(Bennett and Campbell, 2000), the other one by "bisecting the closest points in the convex hull"(Bennett and Campbell, 2000). The solutions of the two different methods are identical and both of the separating hyperplanes can be expressed as a solution of a quadratic program. While the first method explicitly uses the notion of supporting planes, the second one mainly relies on distances between convex hulls. This second method will be used in the thesis to transfer the definition of maximum margin separation to finite closure systems (see, Section 5.3).

Following the work of Boser et al. (1992) and Cortes and Vapnik (1995), many authors have worked on different aspects of maximum margin separation in Euclidean spaces and developed different support vector machines for more general spaces. Since this strongly relates to our adaptation of maximum margin separations to finite closure systems, we also summarize different ideas of generalizing the notion of maximum margins and put them into the context of this thesis.

3.2.1 Maximum margin separations

Hein and Bousquet (2003) generalize SVMs to arbitrary metric spaces by defining maximum margin separations by means of appropriate embeddings (ideally isometric embeddings) of metric spaces into Banach and Hilbert spaces. They show that the SVM algorithm works for all metric spaces that can be embedded into a Hilbert space. Der and Lee (2007) generalize the results on the Euclidean spaces to Banach spaces and Fukumizu et al. (2011) compare learning in Hilbert and in Banach spaces. With the same goal as (Hein and Bousquet, 2003) but with a completely different approach using so-called "Lipschitz Classifiers" instead of linear separators is proposed by von Luxburg and Bousquet (2004) to generalize support vector machines to metric spaces. Other works consider maximal margin separations in fuzzy number spaces (He and Li, 2011) and general metric spaces (Gottlieb et al., 2014).

In contrast to the above methods, we assume a somewhat weaker adaptation. More precisely, we are interested in preserving only some of the properties from Euclidean spaces. Moreover, we are *not* interested in the embedding process itself and work directly with the abstract finite closure systems using a very general proximity definition. This results, for example, in a slightly different definition of maximum margins (Section 5.3).

We assume that elements with high proximity are more likely to be in the same class. However, the structure of the closure system may additionally influence the elements' classes.

Thus, our approach is more related to learning in *distance spaces* (Jain and Obermayer, 2009). As the name suggests, distance spaces are spaces equipped with a distance function, without any further structural assumptions. Maximum margin classifiers in distance spaces are usually called "large-width" or "large-margin" classifiers. They are often based on purely *pairwise similarities*, such as maximum margins in "pseudo-Euclidean spaces" (Graepel et al., 1999). "Large-width" classifiers are intensively studied by Anthony and Ratsaby (2016, 2018, 2020). Similar to our approach, Anthony and Ratsaby (2018) use an adaption of the definition of half-spaces and define maximum margins by the "width" of half-spaces in distance spaces. Nevertheless, this definition differs from ours because it is directly based on the given pairwise similarities between points. In contrast, we are interested in half-spaces defined by the underlying *closure systems*. In particular, our definition of half-spaces is *independent* of the pairwise similarities.

Instead of a metric space or pairwise distances, we use *monotone linkage functions* introduced by Mullat (1976) for defining separation margins. Monotone linkage functions provide a more general measure, by noting that finite closure systems always give rise to monotone linkage functions (see Section 2.3). However, pairwise proximities and metrics need to be defined additionally. This means monotone linkage functions provide a natural and straightforward definition of measuring proximities between sets and elements. They are used in clustering (Kempner and Levit, 2010; Kempner et al., 1997) over set systems and convex geometries. Vashist et al. (2007) apply monotone linkage functions to clustering on graphs and show their practical potential for margin definitions and maximum margin separations of disjoint closed sets.

3.3 Mining and Learning in Finite Closure Systems

This section is devoted to different applications of mining and learning in finite closure systems. Most of the related results are concerned with some vertex classification or clustering task in graphs. Nevertheless, graph convexity has recently been used for other tasks as well, such as, for example, genome rearrangement problems (Cunha and Protti, 2018, 2019), social network analysis (Marc and Šubelj, 2018; Šubelj, 2018; Šubelj et al., 2019), and a new measure of graph centrality (Cerdeira and Silva, 2021). Since Chapter 6 builds up on the work of Marc and Šubelj (2018) and Cerdeira and Silva (2021), we will provide more details about these works in Section 3.3.1 and Section 3.3.2.

Regarding the literature on vertex classification in geodesic closure systems over graphs, we note that there are approaches that deal with the different learning (*supervised*, *online*, and *active learning*) tasks. Regarding supervised learning on graphs, de Araújo et al. (2019) introduce the *geodesic classification* (GC) problem. It is somehow orthogonal to the half-space separation (HSS) and maximum margin separation (MCSS) problems defined and considered in this thesis. In our *supervised* setting, we start with a set of training samples partitioned into two sets according to their labels that have disjoint closures.

We then try to find a *consistent* hypothesis by extending the two sets into two *maximal* disjoint closed sets. In contrast, for the geodesic classification problem, the closures of the two *training* sets may intersect and the task is to remove the outliers, i.e., to find maximum subsets of the initial sets such that their closures do not intersect. Using a less restrictive notion of convexity in metric spaces Stadtländer et al. (2021) introduce learning of *weakly convex* sets. Weak convexity denotes some local convexity in the sense that it consists of the union of ordinary convex hulls of nearby points, where "nearby" is defined by some threshold θ. For example, the g_k-convexity on graphs (Pelayo, 2013) that is based on shortest paths of length at most k (instead of all shortest paths for the geodesic convexity), is a weak convexity on graphs for $\theta = k$.

Thiessen and Gärtner (2021) study *active learning* of half-spaces in geodesic convexity. Again, the starting point is some unknown hypothesis consisting of binary labeled half-spaces. By querying chosen vertices in some "intelligent" way, they try to classify all the graph labels, subject to minimizing the overall number of such queries. Moreover, they provide upper and lower bounds for the number of queries. In a follow-up paper, Thiessen and Gärtner (2022) also study the task of *online learning* of half-spaces in graphs (Thiessen and Gärtner, 2022). The hypothesis space stays the same, but the learner's task is to find a strategy that minimizes the prediction error for the vertices provided by some adversary. The work of Bressan et al. (2021) strongly relates to this two approaches in the sense that it uses oracle queries to recover exact clusters.

Finally, below we briefly discuss two further approaches. We start with the work of Marc and Šubelj (2018) on geodesic core-periphery decompositions, and continue with a new graph centrality measure called graph Tukey depth (Cerdeira and Silva, 2021).

3.3.1 Geodesic Core-Periphery Decompositions

Core-periphery models provide a new tool for analyzing real-world graphs, such as, for example, like social networks. The core-periphery structure has been defined formally by Borgatti and Everett (1999). Regarding geodesic convexity in graphs, a *new* type of *core-periphery* network decomposition has been proposed by Marc and Šubelj (2018). Their result, as well as the ones in the subsequent papers (Šubelj, 2018; Šubelj et al., 2019) clearly demonstrate that geodesic convexity-based core-periphery decomposition provides further valuable insights into the network's structure, which have not been captured before. More precisely, utilizing geodesic convexity, a broad class of real-world networks can be decomposed into a *dense core* surrounded by a *sparse periphery* (see Figure 2.7 for a relatively small example). In contrast to the core, the shortest paths between most node pairs in the periphery are unique. As mentioned above, such a decomposition enables the acquirement of new knowledge about the network at hand (Marc and Šubelj, 2018). For example, the nodes in the core govern the degree distribution of the entire network or they have higher clustering coefficients and a smaller geodesic distance to each other than the periphery nodes. A further interesting property of convexity-based core-periphery decomposition is that it is *not* characteristic for all network types. In particular, while for example social networks typically possess this kind of decomposition, this is not the case for standard random graph and network models, such as the Erdős-Rényi, Watts-Strogatz, or Barabási-Albert models (see Marc and Šubelj, 2018).

3.3.2 Tukey Depth

Tukey depth was originally defined over finite subsets of $\mathbb{R}^d$ (Donoho and Gasko, 1992; Tukey, 1975). It depends only on separating hyperplanes, without taking into account the geometric position of the elements. This property allows for adapting it to other, abstract domains associated with some *abstract closure system* by using (abstract) half-spaces (Chepoi, 1994; Jamison, 1974). For $\mathbb{R}^d$ and in other more general metric spaces (Dai et al., 2022), Tukey depth has been studied in the context of *machine learning*, in particular, for object classification (Mozharovskyi, 2015). In case of learning linear classifiers, the *Tukey median*, i.e., the points with the highest Tukey depth are related to the Bayes point (Gilad-Bachrach et al., 2004). A crucial advantage of Tukey depth over other centrality measures is that the *exact* geometric position of the points in $\mathbb{R}^d$ is *not* relevant for their depth. Thus, Tukey depth is relatively *stable* with respect to outliers and is therefore used also for outlier detection Becker and Gather (1999); Bremner et al. (2008).

Regarding *graphs*, the Tukey depth of a node v of a graph is defined by the size (i.e., number of nodes) of the underlying graph subtracted by the maximum size of a *geodesically closed* set that does *not* contain v itself (Cerdeira and Silva, 2021) (see Figures 2.8 and 6.9 for an example of the graph Tukey depth for different graphs). This definition is closely related to the original definition (Donoho and Gasko, 1992; Tukey, 1975), however, with the difference that in $\mathbb{R}^d$, a half-space with the *maximum* trace on the ground set is used, instead of maximum size geodesic closed sets. Similarly to $\mathbb{R}^d$, it is NP-hard to compute the graph Tukey depth of a node (Cerdeira and Silva, 2021; Johnson and Preparata, 1978).

4 Half-Space Separations in Finite Closure Systems

Linear separations in $\mathbb{R}^d$ by hyperplanes are an *integral* part of classical machine learning algorithms. Since we are interested in mining and learning in *finite closure systems*, one of our primary goals is to transfer linear separation methods from $\mathbb{R}^d$ to finite closure systems by adapting the idea of hyperplane separation. Accordingly, this chapter is devoted to half-space separations in finite closure systems.

Although the adaptation may seem simple at first, several interesting questions arise at a closer look at the problem. More precisely, we have already seen in Chapter 1 that the following fundamental property in $\mathbb{R}^d$ that two disjoint closed sets are always separable by a hyperplane does not hold in general. That means that there exist *abstract* finite closure systems and disjoint closed sets A and B that are *not* separable by half-spaces. To this end, we formulate the Half-Space Separation (HSS) problem for finite closure systems. More precisely, given a finite closure system $(E, \mathcal{C})$ and subsets A and B of E, our goal is to decide if there exist separating half-spaces. We state the formal definition of the concrete problem and show that if the closure operator is computable in time polynomial in the size of E, then the decision problem is NP-complete. Consequently, as we are interested in computationally feasible solutions we need to relax the hardness of the original problem. In particular, we study such closure systems that always allow half-space separations of disjoint closed sets. These systems are called *Kakutani* closure systems. In Section 5.1 we provide an algorithm finding separating half-spaces efficiently for Kakutani closure systems. It is natural to ask whether or not a given closure system has the Kakutani property. We call this problem the Kakutani Problem and show that in the worst case, it requires exponentially many closure operator calls to decide the Kakutani property for a given closure system. However, this result is mainly of theoretical interest, as we consider specific domains for the practical aspects discussed in this thesis. Our particular focus is on characterizations of the Kakutani property for *geodesic* closure systems in graphs. Studying Kakutani closure systems over graphs, we use a characterization result that connects the Kakutani property in graphs with the *Pasch Axiom* (Chepoi, 1994). We can use this characterization result to show that $K_{2,3}$-minor free graphs have the Kakutani property. As a direct consequence, all outerplanar graphs are Kakutani. That is, two disjoint closed vertex sets in outerplanar graphs can always be separated by half-spaces.

Although we mainly focus on binary separations, we consider the more general case of partitioning a closure system into *multiple* closed sets. That is, for a closure system we define the n-Kakutani property and show that in general for some $k, l \in \mathbb{N}$ with $k > l$, neither k-Kakutani implies l-Kakutani nor l-Kakutani implies k-Kakutani.

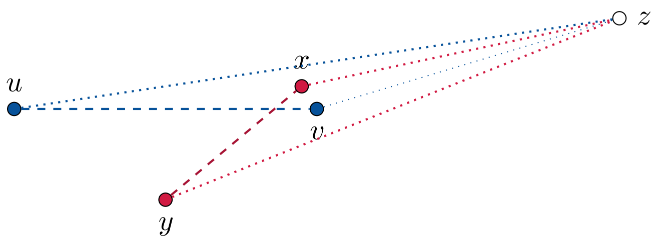


Figure 4.1: Example of a point configuration in $\mathbb{R}^2$, where adding z to either of the closed sets $\{x, y\}$ and $\{u, v\}$ would violate the disjointness condition.

Outline The rest of the chapter is structured as follows. In Section 4.1 we first present the Half-Space Separation (HSS) problem. Subsequently, in Section 4.2 we study different aspects of the Kakutani property. These include the generalization of the (binary) Kakutani property to multiple classes (Section 4.2.1). Finally, considering the domain of graphs, we give some new characterizations of Kakutani closure systems (Section 4.2.2) and conclude in Section 4.3.

4.1 The Half-Space Separation Problem

This section is devoted to half-space separations in finite closure systems. Similarly to the (infinite) closure system over $\mathbb{R}^d$ defined by the family of all convex hulls in $\mathbb{R}^d$, we assume that all finite closure systems $(E, \mathcal{C}_\rho)$ in this section are given *intensionally* by the underlying closure operator ρ. This assumption is natural, as $|\mathcal{C}_\rho|$ can be exponential in $|E|$. We first formulate some results concerning the computational complexity of the following problem:

Problem 4.1.1 (The Half-Space Separation (HSS) Problem)**.** *Let $(E, \mathcal{C}_\rho)$ be a finite closure system given by the corresponding closure operator ρ.* Given *non-empty subsets $A, B \subseteq E$, decide whether A and B are half-space separable in $\mathcal{C}_\rho$, or not.*

For algebraic reasons we disregard the degenerate case of $A = \emptyset$ or $B = \emptyset$. Furthermore, similarly to the infinite closure system over $\mathbb{R}^d$ defined by the family of all convex hulls in $\mathbb{R}^d$, we suppose that the *abstract* closure system is given *implicitly*. More precisely, we assume that $(E, \mathcal{C}_\rho)$ is given by the corresponding closure operator ρ, which returns $\rho(X)$ for any $X \subseteq E$ in *unit* time. The assumption that $\mathcal{C}_\rho$ is given *implicitly* (or *intensionally*) is natural, as $|\mathcal{C}_\rho|$ can be exponential in $|E|$.

Clearly, the solution of an instance of the HSS problem is always "No" whenever $\rho(A) \cap \rho(B) \neq \emptyset$. However, as shown in the example below, the converse of the implication is not true, i.e., the disjointness of the closures of A and B does not imply their half-space separability in $\mathcal{C}$.

Example 4.1.1. *Consider the set $E \subset \mathbb{R}^2$ consisting of the six points in Figure 4.1 and the α-closure system $(E, \mathcal{C}_\alpha)$ defined in Section 2.2.2. Though $\{u, v\}$ and $\{x, y, w\}$ are both closed* *(i.e., belong to $\mathcal{C}_\alpha$) and disjoint, they are not half-space separable in $\mathcal{C}_\alpha$, as z can be added to neither of the sets without violating the disjointness property of half-space separation.*

This difference to $\mathbb{R}^d$ makes, among others, the more general problem setting considered in this work computationally difficult, as shown in Theorem 4.1.1 below. The fact that the disjointness of $\rho(A)$ and $\rho(B)$ does not imply half-space separability of A and B makes the HSS problem computationally intractable. To prove this negative complexity result, we adopt the definition of *convex* vertex sets of a graph defined by shortest paths (Harary and Nieminen, 1981; Mulder, 1980), also referred to as the *geodesic closure*. The geodesic closure operator is denoted by γ and the corresponding closure system is referred to as *geodesic* closure system throughout this thesis (see (2.2) on page 21 for the definition of the geodesic closure system for graphs). Using the definition of graph convexity, we consider the following problem:

Convex 2-Partitioning Problem: *Given* an undirected graph $G = (V, E)$, *decide* whether there is a *proper* partitioning of V into two convex sets.

This problem is known to be NP-complete (Artigas et al., 2011). Notice that the condition on properness is necessary, as otherwise $\emptyset$ and V would always form a (trivial) solution. Note also the difference between the HSS and the Convex 2-Partitioning problems. The latter one is concerned with a property of G (i.e., it has no additional input A, B). We have the following negative result.

Theorem 4.1.1. *The HSS problem is* NP*-hard.*

Proof. Let $G = (V, E)$ be an instance of the Convex 2-Partitioning problem and γ the closure operator corresponding to the closure system as defined in (2.2) on page 21. It holds that G has a proper convex 2-partitioning if and only if there are $u, v \in V$ with $u \neq v$ such that $\gamma(\{u\})$ and $\gamma(\{v\})$ are half-space separable in $(V, \mathcal{C}_\gamma)$. Indeed, if G has a proper convex 2-partitioning then there exist $u, v \in V$ belonging to different convex partitions. Since the two convex partitions are (complementary) half-spaces in $(V, \mathcal{C}_\gamma)$, $\{u\}$ and $\{v\}$ are half-space separable in $(V, \mathcal{C}_\gamma)$. Conversely, if there are $u, v \in V$ such that $\{u\}$ and $\{v\}$ are half-space separable in $(V, \mathcal{C}_\gamma)$, then the corresponding half-spaces form a proper convex 2-partitioning of G. Putting together, the Convex 2-Partition problem can be decided by solving the HSS problem for the input $(V, \mathcal{C}_\gamma)$, $A = \{u\}$, and $B = \{v\}$ for all $u, v \in V$. This completes the proof, as the number of vertex pairs is quadratic in the size of G. □

Note that the negative result above is independent of the time complexity of the closure operator. Below we consider the HSSPoly-problem, which is defined in the same way as the HSS problem, but with the difference that the closure operator ρ corresponding to the closure system $(E, \mathcal{C}_\rho)$ can be computed in $O(p(|E|))$-time for some polynomial p. Theorem 4.1.1 immediately implies the following negative results:

Corollary 4.1.1. *The HSSPoly-problem is NP-complete.*

Proof. The geodesic closure operator γ on graphs can be computed in polynomial time. Hence it follows that HSSPoly can be reduced to the Convex 2-Partitioning problem like HSS as shown above. Moreover, the problem HSSPoly is in NP because for any $A, B, H \subseteq E$, one can verify by definition in $O(p(|E|))$-time, whether H and H^c form a half-space separation of A and B in $\mathcal{C}$, or not. $\square$

The next corollary is concerned with the complexity of computing a closed set separation of *maximum* size. Clearly, it is also a hard problem to decide if a closed set separation of a certain size k exist, see Corollary 4.1.2 below. Note that the case of $k = |E|$ corresponds to the HSS problem.

Corollary 4.1.2. *Let $(E, \mathcal{C}_\rho)$ be a finite closure system given by the corresponding closure operator ρ.* Given *non-empty subsets $A, B \subseteq E$ and an integer $k > 0$, it is NP-hard to decide whether there is a closed set separation $C_1, C_2 \in \mathcal{C}_\rho$ of A, B such that $|C_1| + |C_2| \geq k$.*

4.2 Kakutani Closure Systems

A natural way to overcome the negative results stated in Theorem 4.1.1 and Corollaries 4.1.1 and 4.1.2 is to consider closure systems in which *any* two disjoint closed sets are half-space separable. We will recall the definition of these specific closure systems. More precisely, for a closure operator ρ over a ground set E, the corresponding closure system $(E, \mathcal{C}_\rho)$ is *Kakutani*[1] if it fulfills the (S4) separation axiom defined as follows: For all $A, B \subseteq E$,

$$A \text{ and } B \text{ are half-space separable in } (E, \mathcal{C}_\rho) \iff \rho(A) \cap \rho(B) = \emptyset$$

(see Chepoi (1994) for a good reference on closure systems satisfying the (S4) separation property). By Proposition 2.2.4, any half-space separation of A, B in $\mathcal{C}_\rho$ is a half-space separation of $\rho(A)$ and $\rho(B)$ in $\mathcal{C}_\rho$. Clearly, the HSSPoly problem can be decided in polynomial time for Kakutani closure systems: For any $A, B \subseteq E$ just calculate $\rho(A)$ and $\rho(B)$ and check whether they are disjoint, or not. Furthermore, if A and B are half-space separable, $(E, \mathcal{C}_\rho)$ is Kakutani, and ρ can be computed in time polynomial in n, then Algorithm 1 (see Section 5.1) returns a half-space separation of A, B in polynomial time.

Example 4.2.1. *The closure system used in Example 4.1.1 is not Kakutani. For an example of Kakutani closure systems, consider an arbitrary non-empty finite subset $E \subset \mathbb{R}^2$ of a circle and define the set system $\mathcal{C} \subseteq 2^E$ as follows: For all $E' \subseteq E$, $E' \in \mathcal{C}$ if and only if there exits a closed half-plane $H \subseteq \mathbb{R}^2$ satisfying $E' = H \cap E$. One can easily check that $(E, \mathcal{C})$ is a Kakutani closure system.*

We show in Section 5.1 that Kakutani closure systems guarantee an efficient computation of half-space separations. Hence, it is natural to ask for characterizations of such

[1] A similar property was considered by the Japanese mathematician Shizou Kakutani for Euclidean spaces (cf. Kakutani, 1937)

systems. The first question that arises is whether the Kakutani property for a given closure system can be decided efficiently. More precisely, we are interested in the following problem:

Kakutani Problem: *Given* a closure system $(E, \mathcal{C}_\rho)$, where $\mathcal{C}_\rho$ is given by the corresponding closure operator ρ, *decide* whether $(E, \mathcal{C}_\rho)$ is Kakutani, or not.

Theorem 4.2.1 below answers the question about the efficient detection of Kakutani closure systems negatively by showing that in the worst case we need exponentially many closure operator calls to decide the problem.

Theorem 4.2.1. *Any algorithm solving the Kakutani problem above requires $\Omega\left(2^{|E|/2}\right)$ closure operator calls in the worst case.*

Proof. We can assume without loss of generality that $\emptyset \in \mathcal{C}_\rho$, as otherwise there are no two separable subsets of E. For any even number[2] $n \in \mathbb{N}$ with $n \geq 4$, consider a set E with $|E| = n$ and the set system

$$\mathcal{C}_\rho = \{X \subseteq E : |X| \leq n/2\} \cup \{E\} \ .$$

We claim that $(E, \mathcal{C}_\rho)$ is a Kakutani closure system. Since $\emptyset, E \in \mathcal{C}_\rho$ and $|C_1 \cap C_2| \leq n/2$ for any $C_1, C_2 \in \mathcal{C}_\rho$, $(E, \mathcal{C}_\rho)$ is closed under intersection and hence, it is a closure system. To see that it is Kakutani, notice that all $X \in \mathcal{C}_\rho$ with $|X| = n/2$ are half-spaces; all other closed sets $Y \in \mathcal{C}_\rho$ with $0 < |Y| < n/2$ are not half-spaces. Thus, for any non-empty $A, B \subseteq E$ with $\rho(A) \cap \rho(B) = \emptyset$, $\rho(A)$ can be extended to a half-space $H_1 \in \mathcal{C}_\rho$ such that $H_1 \cap \rho(B) = \emptyset$. By construction, H_1 and its complement H_1^c form a half-space separation of A and B. Hence, $(E, \mathcal{C}_\rho)$ is Kakutani. Note also that for any $C \in \mathcal{C}_\rho$ with $|C| = n/2$, $(E, \mathcal{C}_\rho \setminus \{C\})$ remains a closure system, but becomes non-Kakutani because for any $x \in C$, the disjoint closed sets $\{x\}$ and C^c are not half-space separable.

We are ready to prove the lower bound claimed. Suppose for contradiction that there exists an algorithm $\mathfrak{A}$ that decides the Kakutani problem with strictly less than $\binom{n}{n/2} = \Omega\left(2^{n/2}\right)$ closure operator calls. Then, for $(E, \mathcal{C}_\rho)$ above, there exists a half-space $C \in \mathcal{C}_\rho$ with $|C| = n/2$ such that $\mathfrak{A}$ has not called ρ for C. But then $\mathfrak{A}$ returns the same answer for the Kakutani and non-Kakutani closure systems $(E, \mathcal{C}_\rho)$ and $(E, \mathcal{C}_\rho \setminus \{C\})$, contradicting its correctness. $\square$

The exponential bound in Theorem 4.2.1 is a worst-case bound. Fortunately, there is a broad class of closure systems that are known to be Kakutani. In particular, as a generic application field of Kakutani closure systems we focus on Kakutani closure systems over *graphs* (Section 4.2.2) and on those over finite *lattices* (Section 5.2.2).

Moreover, for the particular class of d-ary closure systems (see Section 2.2.1 for the definition) with efficiently computable closure operators ρ, the above result can be improved by using a result of Chepoi (1994). We note that the result below does not contradict Theorem 4.2.1 above because d can depend on n.

[2]A similar proof applies to odd numbers. For simplicity, we omit the discussion of that case.

Proposition 4.2.1. [3] *Let $(E, \mathcal{C}_\rho)$ be a d-ary closure system with corresponding closure operator ρ and $n = |E|$. The Kakutani Problem can be decided using $n^{O(d)}$ closure operator calls.*

Proof. Following (Chepoi, 1994), the closed set $P_d := \rho(\{x_1, \dots, x_d\})$ with $x_1, \dots, x_d \in E$ is referred to as a *d-polytope*. A d-ary closure system $(E, \mathcal{C}_\rho)$ is Kakutani if and only if $P_{d-1}|P_d := \{x \in E : \rho(P_{d-1} \cup x) \cap P_d \neq \emptyset\}$ is closed for all possible combinations of d and $d-1$ polytopes (Chepoi, 1994, Thm 2.). Obviously, we can compute all possible d (resp. $d-1$) polytopes using $n^{O(d)}$ closure operator calls by considering the closure of all combinations of d (resp. $d-1$) tuples from E. In total there are $O(n^{2d-1})$ different combinations of d and $(d-1)$-polytopes. Hence, all sets, $P_{d-1}|P_d$ can be computed using $n^{O(d)}$ closure operator calls. Finally, to check the Kakutani property, we only need check if $P_{d-1}|P_d$ is closed for all possible combinations of d and $(d-1)$-polytopes. Thus, summarizing all the runtime results, the Kakutani property can be checked using $n^{O(d)}$ closure operator calls. □

Our primary focus in this work is on binary separation. Nevertheless, one can also look at the problem of partitioning the space into *multiple* disjoint closed sets. The following section provides a brief perspective on this generalized problem.

4.2.1 The n-Kakutani property

The Kakutani property of finite closure systems can be considered not only for the binary case, but also for the more general case of n-disjoint closed sets. Accordingly, we call a finite closure system $(E, \mathcal{C}_\rho)$ an *n-Kakutani closure system* if and only if for all n pairwise disjoint closed sets $C_1, \dots, C_n \in \mathcal{C}_\rho$ there exist n pairwise disjoint closed sets $H_1, \dots, H_n$ with $E = \bigcup_{i=1}^{n} H_i \in \mathcal{C}_\rho$ and $C_i \subseteq H_i$ for all $i \in [n]$. Proposition 4.2.2 shows that in general, there is no connection between k and l-Kakutani closure systems for $l \neq k$ neither for $k < l$ nor for $l < k$.

Proposition 4.2.2. *Let $k, l \in \mathbb{N}$ with $k > l \geq 2$. For all possible choices of k and l there exists a closure system that is k-Kakutani but not l-Kakutani. Moreover, for $l = 2$ and $k = 3$ there exist a l-Kakutani closure systems that is not k-Kakutani.*

We give the rather technical proof of the above Proposition 4.2.2 in the Appendix A.

4.2.2 Kakutani Closure Systems over Graphs

Since we are especially interested in the domain of graphs we investigate some further results dealing with the special cases of Kakutani closure systems over *graphs*. The following result provides a characterization of the Kakutani property in *geodesic* closure systems over graphs in terms of the *Pasch axiom* (Chepoi, 1994).

[3] Personal communication with Victor Chepoi.

Theorem 4.2.2. (Chepoi, 1994) *For any finite graph $G = (V, E)$, the geodesic closure system $(V, \mathcal{C}_\gamma)$ is Kakutani if and only if γ fulfills the Pasch axiom, i.e.,*

$$x \in \gamma(\{u, v\}) \wedge y \in \gamma(\{u, w\}) \implies \gamma(\{x, w\}) \cap \gamma(\{y, v\}) \neq \emptyset \tag{4.1}$$

for all $u, v, w, x, y \in V$.

We will use Theorem 4.2.2 to derive two different characterization results of Kakutani closure systems. First we extend the statement of Theorem 4.2.2 to graph structured partitionings (see Proposition 4.2.3) and second we give a characterization of the Kakutani property in terms of forbidden graph minors (see Theorem 4.2.3). As an example we can derive that outerplanar graphs have the Kakutani property. In fact, the characterization result above holds not only for geodesic closure systems, but also for any interval convexity structure (Chepoi, 1994).[4] We note that our result in Theorem 4.2.3 below holds for geodesic closure systems only.

Besides these results which are stated below, note that Theorem 4.2.2 can be turned into a naïve algorithm that decides the Kakutani problem for geodesic closure systems over graphs in $O(n^8)$ time using $O(n^2)$ space by checking the condition in (4.1) for all quintuples of vertices; the complexity of computing γ for any set of vertices is $O(n^3)$ (Dourado et al., 2009). The following more sophisticated algorithm[5] reduces this time complexity to $O(n^5)$ time, using, however, $O(n^4)$ space. Compute first $\gamma(\{u, v\})$ for all $u, v \in V$ and store them in a matrix M. The time and space complexity of this step is $O(n^5)$ and $O(n^3)$, respectively. Using M, for each pair of closed sets we can decide in $O(n)$ time whether or not they are disjoint and store this information in a binary matrix B. This can be done in $O(n^5)$ time and $O(n^4)$ space. Iterating over all quintuples $Q = (u, v, w, x, y) \in V^5$, we can check in constant time from M and B whether Q fulfills (4.1), implying the time and space complexity claimed above.

Graph Structured Partitionings

In Proposition 4.2.3 below we generalize Theorem 4.2.2 to *graph structured* set systems. More precisely, a *graph structured partitioning* (GSP) is a triple $\mathfrak{G} = (S, G, \mathcal{P})$, where S is a finite set, $G = (V, E)$ is a graph, and $\mathcal{P} = \{\text{bag}(v) \subseteq S : v \in V\}$ is a partitioning of S into $|V|$ non-empty subsets (i.e., $\text{bag}(v) \neq \emptyset$, $\bigcup_{v \in V} \text{bag}(v) = S$, and $\text{bag}(u) \cap \text{bag}(v) = \emptyset$ for all $u, v \in V$ with $u \neq v$). The set $\text{bag}(v)$ associated with $v \in V$ is referred to as the *bag* of v. For a GSP $\mathfrak{G} = (S, G, \mathcal{P})$ with $G = (V, E)$, let $\sigma : 2^S \to 2^S$ be defined by

$$\sigma : S' \mapsto \bigcup_{v \in V'} \text{bag}(v) \tag{4.2}$$

[4]An *interval* over a ground set E is a function $I : E \times E \to 2^E$ such that for all $u, v \in E$, $u, v \in I(u, v)$, $I(u, v) = I(v, u)$, and $I(u, w) \subseteq I(u, v)$ for all $w \in I(u, v)$. A set $C \subseteq E$ is I-closed if $I(u, v) \subseteq C$ for all $u, v \in C$. It holds that the set system over E formed by the family of all I-closed subsets of E, also referred to as *interval convexity structure*, is a closure system (see, e.g., Calder, 1971; Chepoi, 1994). Thus, for any graph $G = (V, E)$, $(V, \mathcal{C}_\gamma)$ is a closure system defined by the interval function $I : V \times V \to 2^V$ mapping (u, v) to the set of vertices on *all* shortest paths between u and v.

[5]Personal communication with Victor Chepoi.

with

$$V' = \gamma(\{v \in V : \mathrm{bag}(v) \cap S' \neq \emptyset\})$$

for all $S' \subseteq S$, where γ is the closure operator defined in (2.2) on page 21.

GSPs arise for example in *graph clustering* (see, e.g. Schaeffer, 2007) and *graph partitioning* (see, e.g. Buluç et al., 2016), which play an important role in many practical applications, such as, for example, community network mining.

Proposition 4.2.3. *Let $\mathfrak{G} = (S, G, \mathcal{P})$ be a GSP with $G = (V, E)$. Then σ defined in (4.2) is a closure operator on S. Furthermore, the corresponding closure system $(S, \mathcal{C}_\sigma)$ is Kakutani if and only if γ corresponding to the closure system $(V, \mathcal{C}_\gamma)$ fulfills the Pasch axiom on G.*

Proof. Since $\mathcal{P}$ is a partitioning of S and γ is a closure operator on V, the extensivity and monotonicity of σ are immediate from those of γ. Furthermore, for all $S' \subseteq S$ we have

$$\gamma(\{v \in V : \mathrm{bag}(v) \cap \sigma(S') \neq \emptyset\}) = \gamma(\{v \in V : \mathrm{bag}(v) \cap S' \neq \emptyset\})$$

by the idempotency of γ. Hence, σ is also idempotent, completing the proof of the first claim.

Regarding the second part, note that the function $\varphi : \mathcal{C}_\gamma \to 2^S$ defined by

$$\varphi : V' \mapsto \bigcup_{v \in V'} \mathrm{bag}(v)$$

for all $V' \in \mathcal{C}_\gamma$ is a bijection between $\mathcal{C}_\gamma$ and $\mathcal{C}_\sigma$, satisfying

$$V_1 \cap V_2 = \emptyset \iff \varphi(V_1) \cap \varphi(V_2) = \emptyset$$

for all $V_1, V_2 \in \mathcal{C}_\gamma$. This immediately implies the second claim. □

Notice that any graph $G = (V, E)$ can be regarded as the (trivial) GSP $\mathfrak{G} = (V, G, \mathcal{P})$, where all blocks in $\mathcal{P}$ are singletons with $\mathrm{bag}(v) = \{v\}$ for all $v \in V$. Hence, Theorem 4.2.2 can be regarded as a special case of the proposition above.

Forbidden Minor Characterization

Using the characterization result above, in the theorem below we give a sufficient condition in terms of *forbidden minors* for the Kakutani property of geodesic closure systems. More precisely, as the main contribution of this section, we show that a closure system of a graph is Kakutani whenever the underlying graph does not contain $K_{2,3}$ as a minor. This result may be of some independent interest as well. To state the theorem, we recall that $K_{2,3}$ denotes the complete bipartite graph (V_1, V_2, E) with $|V_1| = 2$ and $|V_2| = 3$. Furthermore, a graph H is a *minor* of a graph G if H can be obtained from G by a sequence of vertex and edge deletions and edge contractions (see, e.g., Diestel, 2012).

Theorem 4.2.3. *For any finite graph $G = (V, E)$, the geodesic closure system $(V, \mathcal{C}_\gamma)$ is Kakutani if G does not contain $K_{2,3}$ as a minor.*

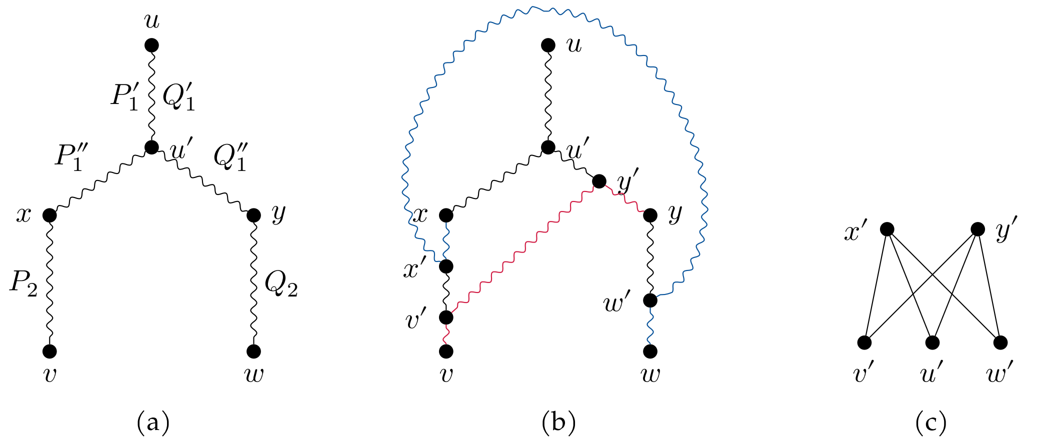


Figure 4.2: Graph minor of non-Kakutani graphs.

Proof. We prove the theorem by contraposition. Let $G = (V, E)$ be a graph such that $(V, \mathcal{C}_\gamma)$ is *not* Kakutani. Then, by Theorem 4.2.2, γ does not fulfill the Pasch axiom, i.e., there are $u, v, w \in V$, $x \in \gamma(\{u, v\})$, and $y \in \gamma(\{u, w\})$ such that

$$\gamma(\{y, v\}) \cap \gamma(\{x, w\}) = \emptyset. \tag{4.3}$$

By definition, $x \in \gamma(\{u, v\})$ (resp. $y \in \gamma(\{u, w\})$) implies that there exists a *shortest* path $P_{uv} = uP_1xP_2v$ between u and v (resp. $P_{uw} = uQ_1yQ_2w$ between u and w), where P_1, P_2 (resp. Q_1, Q_2) denote the (possibly empty) sequences of the interior vertices between u, x and x, v on P_{uv} (resp. u, y and y, w on P_{uw}) (see, also, Figure 4.2(a)).

We claim that u, v, w, x, y are pairwise different. We show this only for x and w; the proofs for the other vertex pairs are similar. Suppose for contradiction that $x = w$. Then $\gamma(\{x, w\}) = \{x\}$. Furthermore, $P_{uw} = uQ_1yQ_2x$ and hence, uP_1x and uQ_1yQ_2x have the same length. Thus, the definition of P_{uv} implies that $uQ_1yQ_2xP_2v$ is a shortest path between u and v. Therefore, yQ_2xP_2v must be a shortest path between y and v. But then $x \in \gamma(\{y, v\}) \cap \gamma(\{x, w\})$, contradicting (4.3).

We are ready to show that (4.3) implies that G contains $K_{2,3}$ as a minor. The definitions of P_{uv} and P_{uw} imply that uP_1x and uQ_1y are shortest paths. Thus, they have a common vertex u' having the maximum distance to u. It follows from (4.3) that $u' \neq x$ and $u' \neq y$, implying that

$$\begin{aligned} P_{uv} &= uP_1xP_2v \ = uP_1'u'P_1''xP_2v \\ P_{uw} &= uQ_1yQ_2w = uQ_1'u'Q_1''yQ_2w \end{aligned}$$

for some P_1', P_1'' (resp. Q_1', Q_1''). We claim that the shortest paths $P_1''xP_2v$ and $Q_1''yQ_2w$ are vertex disjoint. To see this, we distinguish four cases. The case that $P_1''x$ and $Q_1''y$ are vertex disjoint follows directly from the definition of u'. Regarding the case that $P_1''x$ and Q_2w are vertex disjoint, suppose for contradiction that they have a common vertex. But then $x \in \gamma(y, v)$, contradicting (4.3). The argument for the case that P_2v and $Q_1''y$ are vertex disjoint is analogous. Regarding the last case, suppose for contradiction that $P_2 v$ and $Q_2 w$ have a common vertex, say z. Then $xP_2 v$ (resp. $yQ_2 w$) is of the form $xP_2' zP_2'' v$ (resp. $yQ_2' zQ_2'' w$). But then, there are two different shortest paths between u' and z, one through x (i.e., $u'P_1'' xP_2' z$) and one through y (i.e., $u'Q_1'' yQ_2' z$). Accordingly, there exist two different shortest paths between u' and v (i.e., $u'P_1'' xP_2' zP_2'' v$ and $u'Q_1'' yQ_2' zP_2'' v$), and between u' and w (i.e., $u'P_1'' xP_2' zQ_2'' w$ and $u'Q_1'' yQ_2' zQ_2'' w$). Thus, the paths $yQ_2' zP_2'' v$ and $xP_2' zQ_2'' w$ are both shortest paths and contain z, implying $z \in \gamma(\{y,v\}) \cap \gamma(\{x,w\})$, which contradicts (4.3).

Putting together, G contains a subgraph of pairwise vertex disjoint (except for their endpoints) shortest paths as depicted in Figure 4.2(a). Regarding the shortest paths between y and v, note that all of them are vertex disjoint with all shortest paths between u' and x, as otherwise $x \in \gamma(\{y,v\})$ holds by similar arguments as above, contradicting (4.3). Furthermore, again by (4.3), they cannot contain w. In a similar way, all shortest paths between x and w are vertex disjoint with all shortest paths between u' and y, and do not contain v.

Thus, any shortest path P_{yv} between y and v can only have common vertices with the shortest paths P_2 and with at most one of Q_1'' and Q_2 (see the red path in Figure 4.2(b) for an example). Let v' (resp. y') denote the common vertex in the paths P_{yv} and $P_2 v$ (resp. P_{yv} and $Q_1'' yQ_2$) with the maximum distance to v (resp. y). Analogously, any shortest path between x and w can have a common vertex with the shortest path Q_2 and with at most one of P_1'' and P_2 (see the blue path in Figure 4.2(b) for an example). Define w' (resp. x') by the common vertex in the paths P_{xw} and $Q_2 w$ (resp. P_{xw} and $P_1'' xP_2$) with the maximum distance to w (resp. x). Since P_{yv} and P_{xw} are vertex disjoint by (4.3), it follows from the definitions that the shortest paths $P_{u'x'}, P_{u'y'}, P_{v'x'}, P_{v'y'}, P_{w'x'}, P_{w'y'}$ are pairwise disjoint (except for their endpoints) and that the subgraph formed by these six paths contains $K_{2,3}$ as a minor (cf. Figure 4.2(c)), completing the proof of the theorem. □

Remark 4.2.1. *We note that the converse of Theorem 4.2.3 does not hold, implying that $K_{2,3}$ as a forbidden minor does not characterize the Kakutani property for closure systems over graphs. Indeed, for all complete graphs $K_n = (V, E)$, the corresponding closure system $(V, 2^V)$ is Kakutani. The claim then follows by noting that $K_{2,3}$ is a minor of K_n for all $n \geq 5$.*

In Corollary 4.2.1 below we formulate an immediate implication of Theorem 4.2.3. We recall that a graph is *outerplanar* if it can be embedded in the plane such that there are no two edges crossing in an interior point and all vertices lie on the outer face.

Corollary 4.2.1. *For any outerplanar graph $G = (V, E)$, the corresponding geodesic closure system $(V, \mathcal{C}_\gamma)$ is Kakutani.*

Proof. It follows directly from Theorem 4.2.3 and the characterization result that a graph is outerplanar if and only if it contains neither K_4 nor $K_{2,3}$ as minors (Chartrand and Harary, 1967). □

Note that the corollary above applies also to trees, as they are (special) outerplanar graphs. Though the result for trees is well-known, it is typically derived directly from the Pasch axiom. In contrast, we obtain it as an immediate consequence of Theorem 4.2.3.

4.3 Summary

In this chapter we have introduced the half-space separation (HSS) problem, which is one of the central problems for this thesis. It is an adaptation of the linear separation problem in $\mathbb{R}^d$ to that in finite closure systems. Moreover, we derived from the *Convex 2-Partitioning problem* for graphs (Artigas et al., 2011) the *negative* result that the HSS problem is NP-hard for arbitrary and NP-complete for closure systems with corresponding closure operators that can be calculated in polynomial. In particular, our result shows that in contrast to the linear separation problem in $\mathbb{R}^d$, there exists no computationally feasible solution for *arbitrary* finite closure systems. However, the related theoretical result treats the worst-case scenario only and, notably, does not include all types of finite closure systems. Hence, the HSS problem gives rise to studying *special* closure systems and *restrictions* of the problem.

While suitable restrictions of the HSS problem are considered in detail in Chapter 5, this chapter has been devoted to special closure systems. On the one hand, we considered generic structural specializations, on the other hand, we restricted our analysis to the domain of graphs. We note that the special case of geodesic closure systems over graphs will be revisited throughout the entire thesis. Regarding the structural restrictions, we have considered so-called Kakutani closure systems, i.e., closure systems in which disjoint closed sets can be separated by half-spaces. We have shown that under common complexity it cannot be decided in polynomial time if a given closure system is Kakutani or not. Again, this is a worst-case result that can be improved by considering specific closure systems. For d-ary closure systems, we have shown that the problem can be decided using $n^{O(d)}$ closure operator calls. Regarding the particular case of geodesic closure systems over graphs (2-ary closure systems), we have shown that the Kakutani property can be decided in $O(n^5)$ time. The special structure of graphs together with the characterization result of Chepoi (1994) that connects the Pasch axiom with the Kakutani property allows even more. We could show that all graphs that do not contain the complete bipartite graph $K_{2,3}$ as a minor have the Kakutani property.

Finally, since we are interested in applications of the above theory, it is necessary to find some suitable relaxation of the HSS problem. In other words, we are interested in such a relaxed version of the HSS problem that can be solved by polynomial many closure operator calls. At the same time, the problem should preserve as many properties of the original problem as possible. In the next chapter, we present such a relaxation of the HSS problem.

5 Maximal Closed Set Separations in Finite Closure Systems

We have seen that there is neither an efficient solution to the half-space separation (HSS) problem nor we can find separating disjoint closed sets of maximum cardinality efficiently. To resolve these problems, in Section 4.2 we studied closure systems that have the so-called Kakutani property. In this chapter we present efficient separation algorithms for *arbitrary* closure systems. Thus, we need to weaken the HSS problem by relaxing some conditions. More precisely, we give up the *global* optimality of the problem (cf. Theorem 4.1.1 and Corollary 4.1.2) and do not longer require the two separating disjoint sets to be of *maximum* cardinality.

A usual strategy to reduce the hardness of such a problem is to look at *local* optima, instead of *global* ones. Accordingly, we relax the problem definition as follows: Given two disjoint closed sets, find supersets that are closed, disjoint, and *inclusion maximal* with respect to these properties. We refer to this problem as the *maximal closed set separation* (MCSS) problem. To solve this weaker problem, we present a simple *greedy* algorithm. One of the main contributions of this chapter is a theoretical and empirical analysis of this greedy algorithm solving the MCSS algorithm. Besides its simplicity and hence, easy implementation, the algorithm has further advantages. It is a natural question to ask whether there exist *better* algorithms, when the performance is measured by the number of closure operator calls needed to find a solution. We answer this question negatively by showing that in general, there exists *no* algorithm that uses fewer closure operator calls than our simple greedy algorithm. This, somehow, points out the theoretical *optimum* which can be achieved if considering maximal separations in finite closure systems. We stress that the result is primarily of theoretical interest by noting that we are usually confronted with specific closure systems that come with additional structural information in practical use cases. For such specific closure systems, we upgrade our greedy algorithm by showing that it is easily possible to include external domain-specific knowledge that improves theoretical guarantees and practical results.

Regarding the theoretical improvements, we look at the specific case of closure systems over lattices. In fact, there exists a compact representation for closed sets in lattices that is given by two elements. Moreover, we show that the greedy algorithm can be improved in such a way that it only needs logarithmically many closure operator calls in the cardinality of the lattice at hand. Finally, we provide two illustrative examples concerning formal concept lattices (Ganter et al., 2005) and subsumption lattices (Nienhuys-Cheng and de Wolf, 1997) that sketch the semantics of maximal disjoint closed sets in those lattices.

Regarding the practical aspects of our approach, we recall that in ordinary machine learning the Perceptron algorithm (Rosenblatt, 1958) was improved by support vector machines (SVMs) (Boser et al., 1992) by introducing the notion of maximum margin separations. We will adapt this concept and provide a similar technique extending *maximal closed set* separations in finite closure systems to *maximum margin* closed set separations using *monotone linkage functions*. These functions are straightforward generalizations of metrics, i.e., each metric defines a monotone linkage function. We are using this kind of generalization because *abstract* closure systems do not naturally give rise to a metric space. In contrast, as shown in Section 2.3, monotone linkage functions can naturally be defined for finite closure systems without any further information.

Considering a finite closure system equipped with a monotone linkage function, we provide an extended version of our greedy algorithm that incorporates maximum margin separations. This "upgraded" version solves the problem of finding maximum margin separations for disjoint closed sets.

In order to empirically evaluate our methods, we look at the task of binary classification problems over different finite closure systems. In fact, we assume that the domain at hand is binary labeled according to some unknown hypothesis. That is, we only know that the two classes represented by the binary labeled elements are separable by a half-space. Given a subset of these labeled examples for "training", the goal is to predict the remaining *unknown* labels. The specific closure systems we look at are finite point sets in $\mathbb{R}^d$ together with the usual closure operator defined in (2.1) on page 21 and graphs together with the geodesic closure defined in (2.2) on page 21. Finally, we compare the greedy algorithm, i.e., (arbitrary) maximal separations, with the "upgraded" version using maximum margin separations.

Outline The rest of the chapter is organized as follows. In Section 5.1 we present the maximal closed set separation (MCSS) problem and our simple greedy algorithm, and show that it is optimal. In Section 5.2 we concentrate lattices and give an improved version of our original greedy algorithm. We show that, on the one hand, domain-specific knowledge can be easily integrated into our basic algorithm, and, on the other hand, that this knowledge improves the theoretical guarantees. Moreover, we analyze the Kakutani property for closure systems over lattices (Section 5.2.2) and provide two illustrative examples for maximal closed set separations (Section 5.2.3). Section 5.3 is concerned with maximum margin separations. In particular, we introduce the notion of margins for disjoint closed sets (Section 5.3.1) and develop an algorithm that returns a maximum margin separation for some disjoint closed input sets (Section 5.3.2). Evaluating the theory developed, in Section 5.4, we consider the task of binary classification in finite point sets (Section 5.4.1) and graphs (Section 5.4.2). Finally, in Section 5.5, we summarize the results of the chapter.

5.1 The Maximal Closed Set Separation Problem

As shown in the previous chapter, the concept of half-space separations in finite closure systems is somehow "too restrictive" in the sense that a complete covering of the space

Algorithm 1: Maximal Closed Set Separation (MCSS)

```
Input: finite closure system $(E, \mathcal{C}_\rho)$ given by $\rho$ and $A, B \subseteq E$ with $A, B \neq \emptyset$
Output: maximal disjoint closed sets $H_1, H_2 \in \mathcal{C}_\rho$ with $A \subseteq H_1$ and $B \subseteq H_2$ if
        $\rho(A) \cap \rho(B) = \emptyset$; "No" o/w
1  $H_1 \leftarrow \rho(A)$, $H_2 \leftarrow \rho(B)$
2  if $H_1 \cap H_2 \neq \emptyset$ then
3      return "No"
4  $F \leftarrow E \setminus (H_1 \cup H_2)$
5  while $F \neq \emptyset$ do
6      choose $e \in F$ and remove it from $F$
7      if $\rho(H_1 \cup \{e\}) \cap H_2 = \emptyset$ then
8          $H_1 \leftarrow \rho(H_1 \cup \{e\}),\ F \leftarrow F \setminus H_1$
9      else if $\rho(H_2 \cup \{e\}) \cap H_1 = \emptyset$ then
10         $H_2 \leftarrow \rho(H_2 \cup \{e\}),\ F \leftarrow F \setminus H_2$
11 return $H_1, H_2$
```

by two disjoint closed sets is not always possible. Moving away from this restrictive assumption, in this section we propose a relaxation of the HSS problem. In particular, we are interested in such a relaxation that, on the one hand, permits solutions that are efficiently computable, and, on the other hand, provides solutions similar to the original problem. In fact, considering Kakutani closure systems, the solution to the relaxed problem should be a half-space separation. Thus we give up the completeness of the HSS problem by defining the following weaker problem of *maximal* closed set separation.

Maximal Closed Set Separation (MCSS) Problem: Let $(E, \mathcal{C}_\rho)$ be a finite closure system given by the corresponding closure operator ρ. Given non-empty sets $A, B \subseteq E$, *find* a *maximal* closed set separation of A and B in $(E, \mathcal{C}_\rho)$ if it exists; o/w return "NO".

In this section we present Algorithm 1, a simple greedy algorithm solving the MCSS problem and show that it is *optimal* with respect to the number of closure operator calls. Algorithm 1 takes as input a closure system $(E, \mathcal{C}_\rho)$ over some finite ground set E, where $\mathcal{C}_\rho$ is given via the closure operator ρ, and non-empty sets $A, B \subseteq E$. If the closures of A and B are not disjoint, then it returns "NO" (cf. Lines 1–3). Otherwise, the algorithm tries to extend one of the closed sets $C_1 \supseteq A$ and $C_2 \supseteq B$ found so far consistently by an element $e \in F$, where $F = E \setminus (C_1 \cup C_2)$. By consistency we mean that the closure of the extended set must be disjoint with the other (unextended) closed set (cf. Lines 8 and 10). Note that each element of F will be considered at most once for extension (cf. Line 7). If C_1 or C_2 could be extended, then F will correspondingly be updated (cf. Lines 9 and 11), by noting that e will be removed from F even in the case it does not result in an extension (cf. Line 7). The algorithm repeatedly iterates the above steps until F becomes empty; at this stage it returns C_1 and C_2 as a solution.

To state Theorem 5.1.1 concerning some basic properties of Algorithm 1, we introduce the following notation. Let $(E, \mathcal{C})$ be a closure system, $X \subseteq E$, and $C \in \mathcal{C}$ with $X \subseteq C$. Then $\text{gen}(C, X)$ denotes the cardinality of a *smallest* subset of E needed to *generate* C from X, i.e.,

$$\text{gen}(C, X) := \min\{|G| \ : \ G \subseteq E \text{ and } \rho(X \cup G) = C\} \ .$$

Theorem 5.1.1. *Algorithm 1 is correct. Furthermore, for the number N of its closure operator calls it holds that $N = 2$ if $\rho(A) \cap \rho(B) \neq \emptyset$; o/w*

$$\text{gen}(E, A \cup B) + 2 \leq N \leq 2|E| - 2 \ . \tag{5.1}$$

Proof. The correctness is straightforward by noting that the maximality of the output closed sets C_1 and C_2 follows from the monotonicity of ρ, as all elements e considered by the algorithm and not added earlier to one of the closed sets (cf. Lines 9 and 11) can be added later neither to C_1 nor to C_2 without violating the disjointness.

Regarding the second part of the claim, it is trivial for the case that $\rho(A) \cap \rho(B) \neq \emptyset$, so assume $\rho(A) \cap \rho(B) = \emptyset$. For the lower bound in (5.1), note that

$$N \geq 2 + \text{gen}(C_1, A) + \text{gen}(C_2, B) + m \ , \tag{5.2}$$

where $m = |E \setminus (C_1 \cup C_2)|$. We claim that

$$\text{gen}(C_1, A) + \text{gen}(C_2, B) \geq \text{gen}(\rho(C_1 \cup C_2), A \cup B) \ . \tag{5.3}$$

Indeed, by definition there are $G_1, G_2 \subseteq E$ with $|G_1| = \text{gen}(C_1, A)$ and $|G_2| = \text{gen}(C_2, B)$ such that $C_1 = \rho(A \cup G_1)$ and $C_2 = \rho(B \cup G_2)$. Moreover, it holds

$$\begin{aligned} \rho(C_1 \cup C_2) &= \rho(\rho(A \cup G_1) \cup \rho(B \cup G_2)) \\ &= \rho(A \cup B \cup G_1 \cup G_2) \ , \end{aligned} \tag{5.4}$$

where (5.4) is known as the path independence property of closure operators (Monjardet and Raderanirina, 2001). This implies that $|G_1 \cup G_2| \geq \text{gen}(\rho(C_1 \cup C_2), A \cup B)$, from which (5.3) follows by $|G_1| + |G_2| \geq |G_1 \cup G_2|$. We now show that

$$\text{gen}(\rho(C_1 \cup C_2), A \cup B) + m \geq \text{gen}(E, A \cup B). \tag{5.5}$$

By definition, there exists $G \subseteq E$ with $|G| = \text{gen}(\rho(C_1 \cup C_2), A \cup B)$ such that $\rho(C_1 \cup C_2) = \rho(A \cup B \cup G)$. But then, for $X = E \setminus (C_1 \cup C_2)$ we have

$$\begin{aligned} E &= C_1 \cup C_2 \cup X \\ &\subseteq \rho(C_1 \cup C_2) \cup X \\ &= \rho(A \cup B \cup G) \cup X \\ &\subseteq \rho(A \cup B \cup G \cup X) \end{aligned}$$

from which we have (5.5) by $|X| = m$. The lower bound in (5.1) then follows from (5.2)–(5.5).

Regarding the upper bound in (5.1), the algorithm initially calls the closure operator twice (cf. Line 1) and then at most twice per iteration (cf. Lines 9 and 11), giving

$$N \leq 2 \cdot |E \setminus (\rho(A) \cup \rho(B))| + 2 \ .$$

The upper bound then follows from $|\rho(A)|, |\rho(B)| \geq 1$. □

We stress that Algorithm 1 has access to $(E, \mathcal{C}_\rho)$ only via ρ, i.e., it does not utilize any domain-specific properties. The following example shows that, under this assumption, the number of closure operator calls depends on the order of A and B as well as on that of the elements selected in Line 7.

Example 5.1.1. *Let $(E, \mathcal{C}_\rho)$ be the closure system with $E = \{1, 2, \ldots, n\}$ for some integer $n \geq 4$ and with the corresponding closure operator defined by $\rho : X \mapsto \{x \in E : \min X \leq x \leq \max X\}$ for all $X \subseteq E$. Consider first the case that $A = \{2\}$, $B = \{1\}$, and n has been chosen in Line 7 for the first iteration. For this case, the algorithm terminates after the first iteration returning the closed half-spaces $C_1 = \{2, \ldots, n\}$ and $C_2 = \{1\}$, and calling the closure operator together three times.*

Now consider the case that $A = \{1\}$, $B = \{2\}$, and the elements in Line 7 are processed in the order $3, 4, \ldots, n$. One can easily check that the algorithm returns the same two half-spaces after $n-2$ iterations. In each iteration it calls the closure operator twice, giving together the worst-case upper bound $2n-2$ claimed in Theorem 5.1.1.

Though the example above may suggest that Algorithm 1 is not optimal, the upper bound stated in Theorem 5.1.1 is in fact the best possible, regardless of the order of the elements in Line 7 for the general case stated above. Furthermore, the example indicates that using a domain-specific structure might help to reduce the number of closure operator calls drastically. This is shown for lattices in Section 5.2. To show the optimality for the general case, we first prove the following lemma.

Lemma 5.1.1. *All algorithms solving the MCSS problem require at least $2|E|-2$ closure operator calls in the worst case.*

Proof. Suppose for contradiction that there is an algorithm $\mathfrak{A}$ solving the MCSS problem correctly with strictly less than $2|E|-2$ closure operator calls for *all* problem instances. Let $E = \{e_1, \ldots, e_n\}$ for some $n > 2$ and $(E, \mathcal{C}_\rho)$ be the closure system with $\mathcal{C}_\rho = \{\emptyset, \{e_1\}, \{e_2\}, E\}$. We show that there is an element $e \in E \setminus \{e_1, e_2\}$ such that $\mathfrak{A}$ returns the same output for the closure systems $(E, \mathcal{C}_\rho)$ and $(E, \mathcal{C}'_\rho)$ with $\mathcal{C}'_\rho = \mathcal{C}_\rho \cup \{\{e_1, e\}\}$ on input $A = \{e_1\}$ and $B = \{e_2\}$, contradicting its correctness.

By assumption, $\mathfrak{A}$ needs at most $2|E|-3$ closure operator calls to return the unique solution of the MCSS problem for $(E, \mathcal{C}_\rho)$, i.e., the closed sets $\{e_1\}$ and $\{e_2\}$. We claim that $\mathfrak{A}$ needs to calculate the closure of both A and B. Indeed, suppose the closure of one of them, say $\rho(A)$, has not been computed. Then $\mathfrak{A}$ is incorrect, as it would return the same output for the closure systems $(E, \mathcal{C}_\rho)$ and $(E, \mathcal{C}_\rho \setminus \{e_1\})$, though the correct one for $(E, \mathcal{C}_\rho)$ is $\{e_1\}$ and $\{e_2\}$, and "No" for the other one. Thus, $\mathfrak{A}$ can calculate the closure for at most $2|E|-5$ further subsets of E, implying that the closure has not been considered by

$\mathfrak{A}$ for at least one of the sets $\{e_1, e_3\}, \ldots, \{e_1, e_n\}, \{e_2, e_3\}, \ldots, \{e_2, e_n\}$. Let $\{e_1, e\}$ denote such a set. But then, $\mathfrak{A}$ returns the same output for $(E, \mathcal{C}_\rho)$ and $(E, \mathcal{C}_\rho \cup \{e_1, e\})$, although the correct one for $(E, \mathcal{C}_\rho \cup \{e_1, e\})$ (i.e., $\{e_1, e\}$ and $\{e_2\}$) is different from that for $(E, \mathcal{C}_\rho)$; a contradiction. □

Theorem 5.1.2. *Algorithm 1 is optimal with respect to the number of closure operator calls.*

Proof. It is immediate from the upper bound in Theorem 5.1.1 and Lemma 5.1.1. □

In Theorem 5.1.3 below we show that Algorithm 1, besides solving the MCSS problem optimally, also provides an *algorithmic characterization* of Kakutani closure systems.

Theorem 5.1.3. *Let $(E, \mathcal{C}_\rho)$ be a closure system with corresponding closure operator ρ. Then $(E, \mathcal{C}_\rho)$ is Kakutani if and only if for all non-empty $A, B \subseteq E$ with $\rho(A) \cap \rho(B) = \emptyset$, the output of Algorithm 1 is a partitioning of E.*

Proof. The sufficiency is immediate by Theorem 5.1.1 and the definition of Kakutani closure systems. For the necessity, let $(E, \mathcal{C}_\rho)$ be a Kakutani closure system. It suffices to show that for all $e \in F$ selected in Line 7 of Algorithm 1, e is always added to one of H_1 or H_2; the claim then follows by Theorem 5.1.1 for this direction. Suppose for contradiction that there exists an $e \in F$ selected in Line 7 that can be used to extend neither of the closed sets H_1, H_2. Since H_1 and H_2 are disjoint closed sets and $(E, \mathcal{C}_\rho)$ is Kakutani, there are $H_1', H_2' \in \mathcal{C}_\rho$ such that $H_1 \subseteq H_1'$, $H_2 \subseteq H_2'$, and $H_2' = (H_1')^c$. Hence, exactly one of H_1' and H_2' contains e, say H_1'. By the choice of e, $\rho(H_1 \cup \{e\}) \cap H_2 \neq \emptyset$. Since ρ is monotone, $\rho(H_1 \cup \{e\}) \subseteq H_1'$ and hence H_1' and H_2' are not disjoint; a contradiction. □

The characterization result formulated in Theorem 5.1.3 cannot, however, be used to decide in time polynomial in $|E|$, whether a closure system $(E, \mathcal{C}_\rho)$ given *intensionally* by ρ is Kakutani, or not as we need to check the assumption for all possibly exponentially many pairs A, B of disjoint closed sets (cf. Theorem 4.2.1). Besides considering domain specific properties for graphs in Section 4.2.2 we will also consider domain specific properties of lattices regarding maximal closed set separations.

5.2 Closed Set Separations in Lattices

Our second application field is concerned with closure systems over *lattices*. The focus lies, as before, on the HSS and MCSS problems for the special case that the underlying ground set is some finite lattice and the closure operator for a subset S of the ground set is defined by the set of all elements lying between the infimum and supremum of S. For this kind of closure systems we give Algorithm 2, an *adaptation* of Algorithm 1 to lattices. Assuming that the closures of the input sets A and B to be separated are disjoint, Algorithm 2 extends them into a disjoint *maximal ideal* I and a *maximal filter* F such that $A \subseteq I$ and $B \subseteq F$ or vice versa (see Figure 5.1 for an example of different separations of a finite lattice). This specialized version has some important advantages over Algorithm 1. In particular, for certain problem classes it reduces the number of closure operator calls

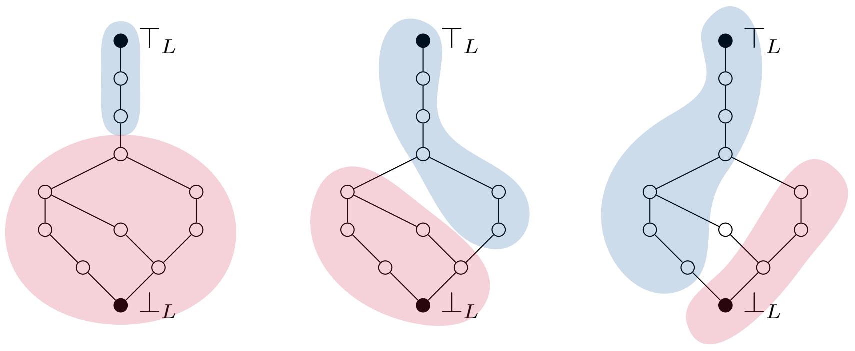


Figure 5.1: The example shows three different separations of the lattice elements $\top_L$ and $\bot_L$ the first two ones are half-space separations while the third one is a maximal disjoint closed set separation. Note that the lattice is not distributive, i.e., our greedy algorithm does not guarantee to find a half-space separation.

logarithmically. This is the situation e.g. in frequent closed itemset mining (Pasquier et al., 1999) or formal concept analysis (Ganter et al., 2005). Furthermore, the disjointness of the closures of any two sets can be decided by comparing their suprema and infima. A further important property of the greedy algorithm specialized to lattices is that it regards the input sets A and B above *symmetrically*. This is a crucial difference e.g. to *inductive logic programming* (see, e.g., Nienhuys-Cheng and de Wolf, 1997), where one is typically interested in finding the smallest ideal of the *subsumption lattices* that contains the set of *positive* examples. If this smallest ideal, with supremum defined by the *least general generalization* (Plotkin, 1970) of the set of positive examples, is *not* disjoint with the set of negative examples, then the separation problem has no solution. This case, however, does not exclude the situation that there is a filter containing the set of positive examples that is disjoint with an ideal containing the set of negative examples. In addition to these properties, we also show that our modified greedy algorithm comprises an *algorithmic* characterization of Kakutani closure systems over lattices. It provides an alternative characterization to the algebraic one that a lattice has the Kakutani property if and only if it is *distributive* (see, e.g., Kubiś, 2002).

5.2.1 Maximal Closed Set Separation in Lattices

Applying Theorem 5.1.1 to λ-closure systems over a lattice $(L; \leq)$, we have that Algorithm 1 requires $O(|L|)$ closure operator calls. If the cardinality of L is exponential in some parameter n, then the bound above becomes exponential in n. As an example, in case of *formal concept analysis* (Ganter et al., 2005), the cardinality of the concept lattice can be exponential in that of the underlying sets of objects and attributes. As another example, the lattice formed by the family of *closed (item)sets* of a transaction database over n items can also be exponential in n (Boros et al., 2003). These and other examples

```
Algorithm 2: Maximal Closed Set Separation in Lattices
  Input: lattice $(L;\leq)$ with $|L| < \infty$ given by an upward and a downward
         refinement operator returning $\mathrm{C}_{\uparrow}(a)$ and $\mathrm{C}_{\downarrow}(a)$ for any $a \in L$, and $A, B \subseteq L$
  Output: supremum of a maximal ideal $I \in \mathcal{C}_{\lambda}$ and infimum of a maximal filter
          $F \in \mathcal{C}_{\lambda}$ separating $A$ and $B$ in $(L, \mathcal{C}_{\lambda})$ with $\lambda$ defined in (2.3) if
          $\lambda(A) \cap \lambda(B) = \emptyset$; "No" o/w
1 if $(\sup A \not\geq \inf B)$ then $\top_I \leftarrow \sup A,\ \bot_F \leftarrow \inf B$
2 else if $(\sup B \not\geq \inf A)$ then $\top_I \leftarrow \sup B,\ \bot_F \leftarrow \inf A$
3 else return "No"
4 while $\exists u \in \mathrm{C}_{\uparrow}(\top_I)$ with $u \not\geq \bot_F$ do $\top_I \leftarrow u$
5 while $\exists l \in \mathrm{C}_{\downarrow}(\bot_F)$ with $l \not\leq \top_I$ do $\bot_F \leftarrow l$
6 return $\top_I, \bot_F$
```

motivate us to adapt the *generic* domain independent Algorithm 1 to lattices, allowing for an upper bound on the number of closure operator calls in terms of the cardinalities of the *upper* and *lower covers* of a lattice and the maximum chain length in L. As we show below, in case of concept lattices or (frequent) closed itemset lattices, the exponential bound above reduces to $O\left(n^2\right)$.

The algorithm solving the MCSS problem for finite lattices is given in Algorithm 2. In case of lattices we assume that the input lattice $(L;\leq)$ is given by an upward $\mathrm{C}_{\uparrow}$ and a downward $\mathrm{C}_{\downarrow}$ refinement operator returning the sets of upper respectively lower covers for the elements of L. For any $A, B \subseteq L$, the algorithm first checks whether their closures are disjoint or not; this is decided by comparing the suprema and infima of A and B (cf. Lines 1–3). If the two closed sets are disjoint then, by Lemma 2.2.2, L has a smallest ideal I and a smallest filter F such that I and F are disjoint and either $\lambda(A) \subseteq I$ and $\lambda(B) \subseteq F$ or vice versa. The algorithm then iteratively tries to extend either I into a larger ideal or F into a larger filter in a way that the extension does not violate the disjointness condition. In the first case, the supremum of I is replaced by one of its upper covers; in the second one the infimum of F by one of its lower covers. Finally, the algorithm stops when any further extension of the current ideal and filter makes them non-disjoint.

Algorithm 2 has some important advantageous properties over Algorithm 1. In particular, while Algorithm 1 considers *all* uncovered elements for the extension of the current closed sets, Algorithm 2 restricts the choice of the next generator element to $\mathrm{C}_{\uparrow}(\sup I) \cup \mathrm{C}_{\downarrow}(\inf F)$, i.e., to a *subset* of the set of elements uncovered so far. Although in the worst case this change does not improve the number of closure operator calls stated in Theorem 5.1.2 for Algorithm 1, below we show that a logarithmic bound holds for certain closure systems over lattices. Another advantageous property of Algorithm 2 is that the disjointness of two closed sets can be decided by comparing two elements only, i.e., the supremum of the current ideal with the infimum of the current filter. Furthermore, the closure operator can be calculated in an easy way by taking advantage of the fact that any closed sublattice of L can be represented by its top and bottom elements. We have the following result for Algorithm 2:

Theorem 5.2.1. *For any λ-closure system over a finite lattice $(L; \leq)$, Algorithm 2 solves the MCSS problem correctly.*

Proof. Let $(L, \mathcal{C}_\lambda)$ be the λ-closure system over a lattice $(L; \leq)$ and $A, B \subseteq L$. The correctness for the case that $\lambda(A) \cap \lambda(B) \neq \emptyset$ (or equivalently, A and B are not separable in $\mathcal{C}_\lambda$) is immediate from Lemmas 2.2.1 and 2.2.2. Applying Lemma 2.2.2 to the case that $\lambda(A) \cap \lambda(B) = \emptyset$, there exist disjoint ideal I and filter F in $\mathcal{C}_\lambda$ such that $A \subseteq I$ and $B \subseteq F$ or vice versa. For symmetry, we can assume without loss of generality that $A \subseteq I$ and $B \subseteq F$. Then, by Lemma 2.2.1, the condition in Line 1 holds and thus, the algorithm terminates in Line 6 for this case. Consider the sequences $u_1, \ldots, u_p \in L$ and $l_1, \ldots, l_q \in L$ selected in this order in Lines 4 and 5, respectively. By construction, $A \subseteq (u_0] \subsetneq (u_1] \subsetneq \ldots \subsetneq (u_p]$ and $B \subseteq [l_0) \subsetneq [l_1) \subsetneq \ldots \subsetneq [l_q)$, where $u_0 = \sup A$ and $l_0 = \inf B$. Furthermore, as $u_p \not\geq l_q$ (cf. Line 5), the ideal $(u_p]$ and filter $[l_q)$ corresponding to the output $\top_I = u_p$ and $\bot_F = l_q$ are disjoint by Lemma 2.2.1. Thus, they form a closed set separation of A and B in $\mathcal{C}_\lambda$.

We now show that $(u_p]$ and $[l_q)$ form a *maximal* closed set separation of A and B in $\mathcal{C}_\lambda$. Suppose for contradiction that there exist $I', F' \in \mathcal{C}_\lambda$ with $I' \cap F' = \emptyset$, $(u_p] \subseteq I'$, and $[l_q) \subseteq F'$ such that at least one of the two containments is proper. We present the proof for $(u_p] \subsetneq I'$ only; the case of $[l_q) \subsetneq F'$ is similar. Since $(u_p] \subsetneq I'$, there exists an $u \in \mathrm{C}_\uparrow(u_p)$ with $u_p \lneq u \leq \sup I'$. But then, by Line 4 we have $u \geq l_q$, which contradicts $I' \cap F' = \emptyset$ by Lemma 2.2.2, as $\sup I' \geq u \geq l_q \geq \inf F'$. □

One can easily check that the number of evaluations of the relation "$\leq$" in Lines 4 and 5 is $O(H_L C_L)$, where H_L is the maximum chain length in L and we denote $C_L := \max_{x \in L} \max\{|\mathrm{C}_\uparrow(x)|, |\mathrm{C}_\downarrow(x)|\}$. Proposition 5.2.1 below utilizes this property for the special case that the underlying lattice is a family of subsets of some finite ground set.

Proposition 5.2.1. *Let $(L, \mathcal{C}_\lambda)$ be a λ-closure system over a lattice $(L; \subseteq)$ with $L \subseteq 2^E$ for some ground set E of cardinality n. Then Algorithm 2 solves the MCSS problem for the λ-closure system over $(L; \subseteq)$ with $O(n^2)$ evaluations of the subset relation.*

Proof. It follows directly from the remark above by $H_L = O(n)$ and $H_C = O(n)$. □

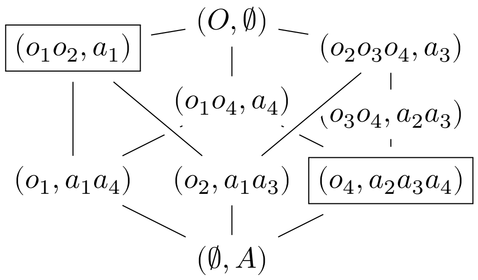
Since $L \subseteq 2^E$ for some ground set E with $|E| = n$, $|L|$ can be exponential in n. As an application of Proposition 5.2.1 to concept lattices and closed (frequent) itemsets, we have that maximal closed separations can be found in time polynomial in the size of the underlying ground sets for these types of closure systems. In Section 5.2.3 we present two illustrative examples of an application of Algorithm 2 to *learning* in formal concept analysis and in first-order logic.

5.2.2 Kakutani Closure Systems over Lattices

In this section we consider *Kakutani* λ-closure systems over finite lattices. As mentioned above, this kind of closure systems have a well-known *algebraic* characterization in terms of distributivity (see, e.g., Kubiś, 2002; van de Vel, 1984). As an orthogonal result, in

	a_1	a_2	a_3	a_4
o_1	1	0	0	1
o_2	1	0	1	0
o_3	0	1	1	0
o_4	0	1	1	1

(a) Binary matrix M over $O \times A$.



(b) Concept lattice defined by M.

Figure 5.2: Example of a formal context (LHS) and its corresponding concept lattice (RHS).

Theorem 5.2.2 below we show that Algorithm 2 provides an *algorithmic* characterization of Kakutani λ-closure systems over lattices.

Theorem 5.2.2. *Let $(L, \mathcal{C}_\lambda)$ be the λ-closure system over a finite lattice $(L; \leq)$. Then $(L, \mathcal{C}_\lambda)$ is Kakutani if and only if for all non-empty $A, B \subseteq L$ with $\lambda(A) \cap \lambda(B) = \emptyset$, for the output $\top_I$ and $\bot_F$ of Algorithm 2 it holds that $(\top_I]$ is a prime ideal, $[\bot_F)$ is a prime filter, and $(\top_I]$ is the complement of $[\bot_F)$.*

Proof. The sufficiency is immediate by Theorem 5.2.1. For the necessity, let $(L, \mathcal{C}_\lambda)$ be a Kakutani closure system and $A, B \subseteq L$ with $\lambda(A) \cap \lambda(B) = \emptyset$. Let $u_1, \dots, u_p$ and $l_1, \dots, l_q$ be the maximal sequences considered in the proof of Theorem 5.2.1 for the case of $\lambda(A) \cap \lambda(B) = \emptyset$. For their last elements we have that $(u_p], [l_q) \in \mathcal{C}_\lambda$ and $(u_p] \cap [l_q) = \emptyset$. Since $\mathcal{C}_\lambda$ is Kakutani, there is a proper partitioning $H, H^c \in \mathcal{C}_\lambda$ of L such that $(u_p] \subseteq H$ and $[l_q) \subseteq H^c$. Thus, $\bot_L \in H$ and $\top_L \in H^c$, implying that H is a prime ideal and H^c is its complement prime filter. Suppose for contradiction that one of the two containments above, say the first one, is proper (i.e., $(u_p] \subsetneq H$). But then, there exists an element $u \in \mathrm{C}_\uparrow(u_p)$ with $u_p \lneq u \leq \sup H$, i.e., u will be selected after u_q in Line 4. This contradicts that u_q is the last element selected in Line 4. $\square$

Corollary 5.2.1. *Let $(L, \mathcal{C}_\lambda)$ be the λ-closure system over a finite lattice $(L; \leq)$. Then $(L, \mathcal{C}_\lambda)$ is distributive if and only if for all non-empty $A, B \subseteq L$ with $\lambda(A) \cap \lambda(B) = \emptyset$, the ideal $(\top_I]$ and filter $[\bot_F)$ defined by the output $\top_I$ and $\bot_F$ of Algorithm 2 form a partitioning of L.*

Proof. Immediate from Theorem 5.2.2 and the characterization of λ-closure systems over finite lattices in terms of distributivity (Kubiś, 2002; van de Vel, 1984). $\square$

5.2.3 Illustrative Examples

We finish this section by presenting two different illustrative examples of our algorithm developed for lattices.

Maximal closed set separation in formal concept lattices Our first example is concerned with *concept lattices* (Ganter et al., 2005) that arise from some formal context (see, Section 2.1.2 for the definition). For our example we consider the concept lattice presented in Figure 5.2(b)[1] that is defined by the formal context $O \times A$, i.e., the binary matrix M (see Figure 5.2(a)), where $O = \{o_1, \dots, o_4\}$ is the set of objects representing

$$\begin{aligned} o_1 &= \text{‘equilateral triangle’},\\ o_2 &= \text{‘right triangle’},\\ o_3 &= \text{‘rectangle’},\\ o_4 &= \text{‘square’}, \end{aligned}$$

and $A = \{a_1, \dots, a_4\}$ is the set of attributes corresponding to

$$\begin{aligned} a_1 &= \text{‘has 3 vertices’},\\ a_2 &= \text{‘has 4 vertices’},\\ a_3 &= \text{‘has a right angle’},\\ a_4 &= \text{‘is equilateral’}. \end{aligned}$$

Suppose we would like to separate the two set of concepts $C_1 = \{(o_4, a_2a_3a_4)\}$ and $C_2 = \{(o_1o_2, a_1)\}$, i.e., the set consisting of the concept ‘square’ from that containing the concept of ‘triangle’ using maximal disjoint closed sets. As $\sup C_1 = (o_4, a_2a_3a_4) \not\geq (o_1o_2, a_1) = \inf C_2$, Algorithm 2 first extends the smallest ideal $I = \{(\emptyset, A), (o_4, a_2a_3a_4)\}$ containing C_1 into a maximal ideal I' such that $\sup I' \not\geq (o_1o_2, a_1)$. Since both elements of $\mathrm{C}_\uparrow((o_4, a_2a_3a_4)) = \{(o_1o_4, a_4), (o_3o_4, a_2a_3)\}$ satisfy this condition, one of them is selected arbitrarily in Line 4, say (o_1o_4, a_4). For the new ideal I with $\sup I = (o_1o_4, a_4)$ we have that it cannot be extended, as for the only element $(O, \emptyset)$ in the upper covers of (o_1o_4, a_4) we have $(O, \emptyset) > (o_1o_4, a_4)$. The algorithm therefore continues in Line 5 by extending the smallest filter $F = \{(o_1o_2, a_1), (O, \emptyset)\}$ containing C_2 into a maximal filter F' such that $\inf F' \not\leq \sup I = (o_1o_4, a_4)$. The only element it can select from $\mathrm{C}_\downarrow((o_1o_2, a_1)) = \{(o_1, a_1a_4), (o_2, a_1a_3)\}$ is (o_2, a_1a_3), as $(o_1, a_1a_4) < (o_1o_4, a_4)$. For the new filter F with $\inf F = (o_2, a_1a_3)$ we have that it cannot be extended, as for the only element $(\emptyset, A)$ in the lower covers of (o_2, a_1a_3) we have $(\emptyset, A) < (o_1o_4, a_4)$. The algorithm therefore terminates with the supremum (o_1o_4, a_4) and infimum (o_2, a_1a_3) of the maximal separating closed sets

$$I = \{(\emptyset, A), (o_1, a_1a_4), (o_4, a_2a_3a_4), (o_1o_4, a_4)\}$$

[1]See (Ignatov, 2015) for more details on this example.

and

$$F = \{(o_2, a_1a_3), (o_1o_2, a_1), (o_2o_3o_4, a_3), (O, \emptyset)\} \ ,$$

respectively. That is, 'square' is separated from 'triangle' by the maximal disjoint closed sets of concepts specializing 'equilateral objects' and generalizing 'right triangles'. Note that I and F do not form a half-space separation because $(o_3o_4, a_2a_3) \notin I \cup F$.

Maximal closed set separation in subsumption lattices Our second illustrative example of the application of Algorithm 2 to lattices is concerned with finding consistent hypotheses in *inductive logic programming* (see, e.g., Nienhuys-Cheng and de Wolf, 1997). For simplicity, the example below is restricted to a very simple first-order vocabulary by noting that the same idea holds for any finite sublattice of a *subsumption lattice* (cf. Nienhuys-Cheng and de Wolf, 1997, for the definition and some formal properties of subsumption lattices). More precisely, in the example below we assume that the vocabulary consists only of a single predicate symbol P of arity n and a set V of variables.

Now consider the subsumption lattice $(L; \leq)$ as defined in Section 2.1.2 for some predicate P with arity $n = 5$ and a set of variables $V = \{v, w, x, y, z\}$ (see, Figure 5.3). For simplicity, we use a "canonical" atom for representing the class of equivalent atoms. We can assume without loss of generality that $\bot_L = P(v, v, v, v, v)$ and $\top_L = P(v, w, x, y, z)$. Let

$$\begin{aligned} A &= \{P(v, w, x, x, z), P(v, w, x, y, y)\} \\ B &= \{P(v, v, v, v, z), P(v, v, v, y, v)\} \end{aligned}$$

denote the sets of positive and negative examples, respectively. In the most common problem setting in inductive logic programming (Nienhuys-Cheng and de Wolf, 1997), one is interested in finding a P-atom $g \in L$ such that g generalizes all positive examples in A and none of the negative examples in B, if such a g exists. Clearly, such a g exists if and only if $\sup A = P(v, w, x, y, z)$ does not generalize any of the P-atoms in B. This is not the case in our example, as $\sup A = P(v, w, x, y, z)$ generalizes both elements in B.[2] Thus, the consistent hypothesis finding problem has no solution. If, however, we only require A and B to be separable in $(L, \mathcal{C}_\lambda)$, then Algorithm 2 returns a solution. Indeed, while

$$\sup A = P(v, w, x, y, z) \geq P(v, v, v, v, v) = \inf B \ ,$$

for $\sup B$ and $\inf A$ we have

$$\sup B = P(v, v, v, y, z) \not\geq P(v, w, x, x, x) = \inf A \ ,$$

implying $\top_I = P(v, v, v, y, z)$ and $\bot_F = P(v, w, x, x, x,)$ for Lines 1 and 2 of Algorithm 2, which, in turn, are extended in Lines 4 and 5 into $\top_I = P(v, v, x, y, z)$ and

[2]In inductive logic programming, $\sup A$ is referred to as the *least general generalization* of A. It can be calculated by Plotkin's algorithm in (Plotkin, 1970).

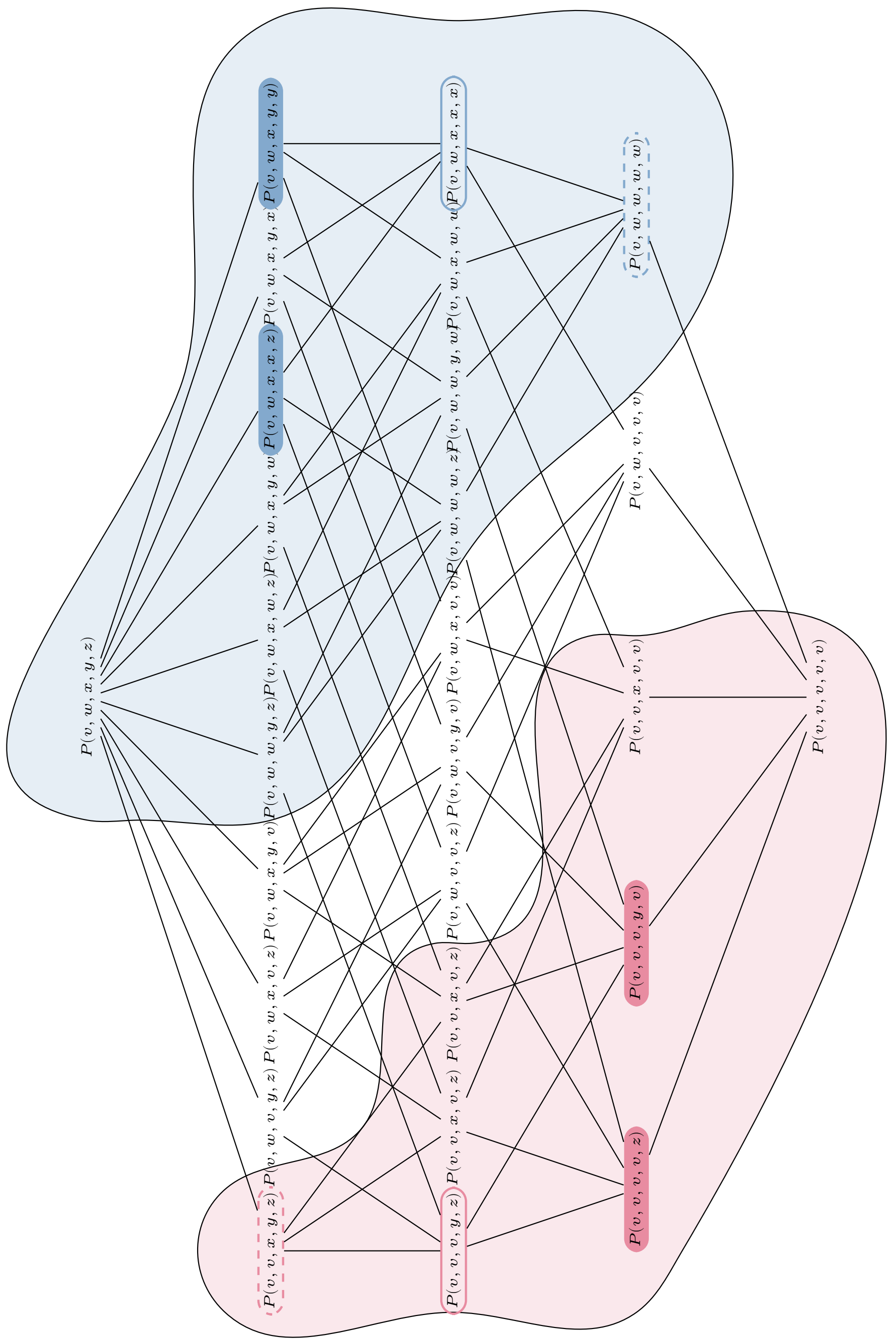


Figure 5.3: Maximal closed set separation of set A (marked by blue) and set B (marked by red) in a subsumption lattice. The elements with bold borders are the supremum of B respectively the infimum of A. The elements with a dashed border are the output elements $\perp_F$ respectively $\top_I$.

$\perp_F = P(v,w,w,w,w)$ (see, also, Figure 5.3). For the corresponding ideal $(P(v,v,x,y,z)]$ and filter $[P(v,w,w,w,w))$ returned by Algorithm 2 we have that they are disjoint, contain B and A, respectively, and are *maximal* in $(L, \mathcal{C}_\lambda)$ with respect to these properties. In other words, the output of the algorithm separates A and B by the sets of P-atoms that are generalizations of $P(v,w,w,w,w)$ and are generalized by $P(v,v,x,y,z)$, respectively. This example also shows that our approach is able to produce an output for such cases where traditional approaches based on Plotkin's least general generalization (Plotkin, 1970) have no solution. The reason is that Algorithm 2 treats the input two sets symmetrically, in contrast to all such approaches.

5.3 Maximum Margin Separations

If the separating maximal closed sets or half-spaces for some disjoint input sets are not unique, Algorithm 1 returns one of them selected arbitrarily. This behavior is similar to Rosenblatt's perceptron algorithm (Rosenblatt, 1958), which also has the only requirement that the output hyperplane has to separate the input point sets. A significant drawback of such unconstrained solutions is that they provide no control of *overfitting*. This problem has been addressed by Vapnik and his co-authors' work on *support vector machines* (SVMs) (Boser et al., 1992), which have become a well-established tool within machine learning for their well-founded theory and excellent predictive performance on a broad range of real-world problems. In particular, support vector machines resolve the problem of overfitting by (implicitly) embedding the data points into an inner product feature space and separating their images in that feature space by a hyperplane maximizing the minimum of the distances to the sets of positive and negative examples.

Motivated by the same problem as in the SVM, in this section, we adapt the idea of maximum margin hyperplanes to binary separation problems over finite closure systems. This adaptation raises, however, several issues. Most notably, while in SVMs, the inner product induces a distance, in case of finite closure systems the ground set is typically *not* a metric space. To overcome this problem, we assume that the closure systems are provided by some *weak* proximity measure defined by *monotone linkage functions* (Mullat, 1976). A closure system equipped with a monotone linkage function is called *monotone linkage closure system* (MLCS) (see Section 2.3). We motivate our choice of proximity measure by recalling that monotone linkage functions, in contrast to metrics, can naturally be defined for a closure system (see Section 2.3). While these kind of functions strongly generalize distance functions (e.g., they are not required to fulfill symmetry or the triangle inequality), they preserve the *anti-monotonicity* of distances. That is, the linkage from a point to a set is anti-monotonic for set inclusion. Similarly to SVMs, this feature is essential for the separation problems considered in this work.

A second issue is how to define margins for closed set and half-space separations in finite closure systems. While there are different equivalent characterizations of maximum margins for SVMs, it turns out that their equivalence does not hold when adapting them to abstract closure systems equipped with monotone linkage functions. In particular, in contrast to SVMs, the linkage of the set of positive examples to a half-space can be different from that of the negative examples to the complementary half-space for *all* half-spaces. We therefore define the *margin* by the smallest linkage from the closures of the input sets to the complementary half-spaces. Furthermore, we generalize this concept to arbitrarily closed set separations as well.

Using these notions, we then formulate the computational problems of finding closed set and half-space separations *maximizing* the margin in finite closure systems equipped with monotone linkage functions. This problem adapts several key features of SVMs to abstract closure systems. For the closed set separation problem, we then give a extended version of Algorithm 1 and show that it is correct and requires a *linear* number of closure operator calls and linkage function evaluations. We also show that for Kakutani closure systems, the algorithm always returns a half-space separation of the input sets with *maximum margin* if and only if the closures of the two training sets are disjoint.

5.3.1 Maximum Margin Separations in Monotone Linkage Closure Systems

As mentioned before, our main goal in this section is to adapt Vapnik's idea (Boser et al., 1992) of *maximum margin* separating hyperplanes to finite closure systems. That is, given subsets A and B of some inner product (feature) space $\mathcal{F}$, in case of *support vector machines* (SVM) (Boser et al., 1992) we are interested in the hyperplane H^* having maximum distance to the two sets, i.e., which satisfies

$$d(A \cup B, H) \leq d(A \cup B, H^*) \tag{5.6}$$

for all hyperplanes H, where for all $X, Y \subseteq \mathcal{F}$, $d(X,Y) = \min_{y \in Y} d(X,y)$ with d being the distance induced by the underlying inner product. It is a well-known fact that if A and B are separable by a hyperplane, then H^* is *unique*; H^* is also referred to as the *maximum margin separating hyperplane*, where the *margin* of a separating hyperplane H is defined by

$$\mu(A,B) = d(A,H) + d(B,H) \ . \tag{5.7}$$

A key property of the margin is that it is *anti-monotone* with respect to set inclusion, i.e., $\mu(A',B') \leq \mu(A,B)$ for all $A' \supseteq A$ and $B' \supseteq B$. Note that (5.6) implies

$$d(A,H^*) = d(B,H^*) \ .$$

Clearly, the above definitions are *not* (directly) applicable to maximum margin separation in closure systems because we do not assume E to be an inner product or a metric space and have therefore no measure in general for the distance from a point $e \in E$ to a subset $X \subseteq E$. Furthermore, while the notion of half-spaces in $\mathbb{R}^d$ has been generalized to closure systems, for hyperplanes there is no analogous definition. Hence, to be in a position to define margins, we need some suitable functions for the abstraction of "closeness" from a point to a subset of the ground set. They should *generalize* metrics, but *preserve* the anti-monotonic property above at the same time.

The class of *monotone linkage functions* (Mullat, 1976) defined in Section 2.3 fulfill both of these requirements. In addition to generality and anti-monotonicity, they have some further properties making this class an attractive candidate for our purpose. In particular, monotone linkage functions assume neither symmetry nor the triangle inequality. Moreover, they can be naturally defined using the closure systems (see Section 2.3).

To adapt the ordinary definition of margins to closure systems equipped with a monotone linkage function (MLCS), note that if a hyperplane $H \subseteq \mathbb{R}^d$ separates A and B, then (5.7) is equivalent to

$$\begin{aligned} \mu(A,B) &= d(A,H_2) + d(B,H_1) \\ &= d(\text{conv}(A),H_2) + d(\text{conv}(B),H_1) \ , \end{aligned} \tag{5.8}$$

where $H_1 \supseteq A$ and $H_2 \supseteq B$ are the closed half-spaces defined by H (i.e., $H \subseteq H_1, H_2$). That is, in case of SVM, the margin given by a hyperplane H separating A and B is defined by the sum of the distances from the *convex hull* of A to the half-space H_2 containing B and from that of B to H_1 containing A.

Analogously to distances in metric spaces, we first extend linkage functions from sets to elements to those from sets to sets. Formally, for a linkage function l on E and subsets $X, Y \subseteq E$, we define the linkage l from X to Y by $l(X,Y) = \min_{y \in Y} l(X,y)$. Note that this extended definition preserves anti-monotonicity, i.e., $l(X',Y) \leq l(X,Y)$ holds whenever $X' \supseteq X$. Let H, H^c be half-spaces of an MLCS $(E, \mathcal{C}_\rho, l)$ and $A \subseteq H, B \subseteq H^c$ for some $A, B \subseteq E$. Then, by analogy with (5.8), our *first* definition of the *margin* of the half-space separation of A, B by H, H^c is

$$\mu^+_{H,H^c}(A,B) = l(\rho(A),H^c) + l(\rho(B),H). \tag{5.9}$$

While the above adaptation of the ordinary notion of margins to MLCSs is relatively natural, the generalization is less obvious for *maximum* margin half-space separations. This is because for SVM there are two *equivalent* properties characterizing maximum margin hyperplanes H^* defining the closed half-spaces $H_1 \supseteq A$ and $H_2 \supseteq B$:

(i) H^* maximizes $\mu(A,B)$ such that $d(\text{conv}(A),H_2) = d(\text{conv}(B),H_1)$.

(ii) H^* maximizes $\min\{d(\text{conv}(A),H_2), d(\text{conv}(B),H_1)\}$.

That is, the maximum margin hyperplane by (i) lies in the "middle" between the convex hulls of A and B; by (ii) it maximizes the minimum of the distances from the two convex hulls. While (i) and (ii) are equivalent in case of SVM, the situation is different for MLCSs as shown in the proposition below.

Proposition 5.3.1. *There exists an MLCS $(E, \mathcal{C}_\rho, l)$ and subsets $A, B \subseteq E$ such that*

$$\mu^+_{H_1,H_1^c}(A,B) \neq \mu^+_{H_2,H_2^c}(A,B),$$

where

$$\begin{aligned} H_1 &= \operatorname*{arg\,max}_{H,H^c \in \mathcal{C}_\rho} \mu^+_{H,H^c}(A,B) \textit{ subject to } l(\rho(A),H^c) = l(\rho(B),H) \\ H_2 &= \operatorname*{arg\,max}_{H,H^c \in \mathcal{C}_\rho} \min\{l(\rho(A),H^c), l(\rho(B),H)\} \ . \end{aligned}$$

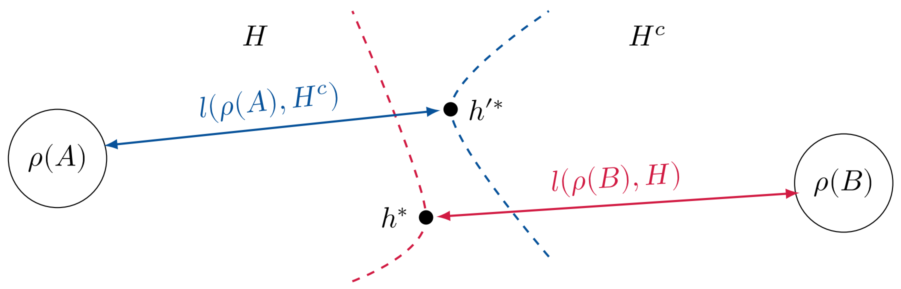


Figure 5.4: Linkages $l(\rho(A), H^c)$, $l(\rho(B), H)$ between closed sets $\rho(A), \rho(B)$ and half-spaces H, H^c together with the support elements h^*, h'^*.

Proof. Consider the MLCS $(E, \mathcal{C}, l)$ with $E = \{a, b, c, d\}$ and $\mathcal{C} = \{X \subseteq E : |X| \neq 3\}$. The monotone linkage function is defined by

$$\begin{aligned} l(\{a\}, b) = l(\{b\}, a) = l(\{b\}, d) &= 3, \\ l(\emptyset, e) &= 3 \text{ for all } e \in E, \\ l(\{a\}, c) &= 2, \\ l(\{a\}, d) = l(\{b\}, c) &= 1, \\ l(X, e) &= 0 \text{ for all other } X \subseteq E \text{ and } e \in E \end{aligned}$$

It can be easily checked that $(E, \mathcal{C})$ is a closure system and l fulfills the anti-monotonicity property. For $A = \{a\}, B = \{b\}$ there exist exactly two different separating half-spaces of size 2, i.e., $H_1 = \{a, c\}$ and $H_2 = \{a, d\}$. Using the definition of linkage on sets it follows $l(A, H_1^c) = l(B, H_1) = 1$. Moreover, $l(A, H_2^c) = 2$ and $l(B, H_2) = 3$. Thus, H_1 fulfills the first property and H_2 the second one, by noting that $2 = \min\{l(A, H_2^c), l(B, H_2)\} > \min\{l(A, H_1^c), l(B, H_1)\} = 1$. The claim then follows by $2 = \mu^+_{H_1, H_1^c}(A, B) \neq \mu^+_{H_2, H_2^c}(A, B) = 5$. □

Thus, for an MLCS $(E, \mathcal{C}, l)$, maximizing the margin as defined in (5.9) subject to $l(\rho(A), H^c) = l(\rho(B), H)$ is *not* equivalent to maximizing

$$\mu_{H, H^c}(A, B) := \min\{l(\rho(A), H^c), l(\rho(B), H)\} \tag{5.10}$$

over *all* half-space separations of A and B in $(E, \mathcal{C}_\rho, l)$ (see, also, Figure 5.4).

Since our primary interest is in classification, we prefer the definition in (ii) above, and will accordingly focus on maximizing the margin defined by (5.10). Note that our definition of margin differs from that in SVM, as it involves only one part of the ordinary one.

Until now we have concentrated on half-space separations. In case of MLCSs, two sets with disjoint closures are, however, not always half-space separable. Fortunately, the above definition of margin can be extended naturally to arbitrary closed sets. More precisely, for an MLCS $(E, \mathcal{C}_\rho, l)$, let $A, B \subseteq E$ and $C_A, C_B \in \mathcal{C}_\rho$ with $A \subseteq C_A$ and $B \subseteq C_B$. Then the *margin* for C_A and C_B is defined by

$$\mu_{C_A, C_B}(A, B) := \min\{l(\rho(A), C_A^c), l(\rho(B), C_B^c)\} \ . \tag{5.11}$$

Similarly to half-spaces, the definition takes only one part of the effective margin into account. Note that (5.10) is the special case of (5.11) for $C_A = H$ and $C_B = H^c$. We now show that the anti-monotonicity of monotone linkages extends to margins in MLCS. This property is essential for separations.

Lemma 5.3.1. *Let $(E, \mathcal{C}_\rho, l)$ be an MLCS, $A \subseteq A' \subseteq E$, $B \subseteq B' \subseteq E$, and $C_A \supseteq A'$, $C_B \supseteq B'$ disjoint closed sets. Then $\mu_{C_A,C_B}(A, B) \geq \mu_{C_A,C_B}(A', B')$.*

Proof. This follows directly from the definition of the margin between two sets in (5.11) and the anti-monotonicity of monotone linkage functions. □

Moreover, maximizing the disjoint closed sets C_A and C_B in Lemma 5.3.1 maximizes the margin at the same time, as we show in the following lemma.

Lemma 5.3.2. *Let $C_A \subseteq C'_A$ and $C_B \subseteq C'_B$ be closed sets of an MLCS $(E, \mathcal{C}_\rho, l)$ with $C'_A \cap C'_B = \emptyset$ and $A \subseteq C_A$, $B \subseteq C_B$. Then $\mu_{C_A,C_B}(A, B) \leq \mu_{C'_A,C'_B}(A, B)$.*

Proof. From the definition of monotone linkages between sets it follows that $l(X, Y) \geq l(X, Y')$ whenever $Y \subseteq Y'$. Hence, by $C^c_A \supseteq C'^c_A$ and $C^c_B \supseteq C'^c_B$ we have

$$\begin{aligned}\mu_{C_A,C_B}(A, B) &= \min\{l(\rho(A), C^c_A), l(\rho(B), C^c_B)\} \\ &\leq \min\{l(\rho(A), C'^c_A), l(\rho(B), C'^c_B)\} \\ &= \mu_{C'_A,C'_B}(A, B) \ .\end{aligned}$$

□

Given a half-space separation of A, B with $A \subseteq H$ and $B \subseteq H^c$, similarly to support vectors in SVMs we can define the *support elements* by h^* and h'^* satisfying $l(\rho(A), H^c) = l(\rho(A), h'^*)$ and $l(\rho(B), H) = l(\rho(B), h^*)$, respectively. For example, in case of maximum margin separating half-spaces in trees, there are exactly two support elements corresponding to the two half-spaces. We have marked the support elements in Figure 5.4, Figure 5.5, and Figure 5.6 by dashed circles.

5.3.2 The Maximum Margin Algorithm

Using (5.10) and (5.11) for the definition of margins for half-space and closed set separations, we are ready to formulate the separation problems in MLCS $(E, \mathcal{C}_\rho, l)$:

Maximum Margin Half-Space Separation (MMHSS) Problem: *Given* non-empty subsets A, B of E, *find* a half-space $H \in \mathcal{C}_\rho$ with $A \subseteq H$, $B \subseteq H^c$ that maximizes the margin, i.e., $H = \underset{\{H_1 \in \mathcal{C}_\rho : H_1^c \in \mathcal{C}_\rho\}}{\arg\max} \mu_{H_1,H_1^c}(A, B)$, if A and B are half-space separable; o/w return "No".

Maximum Margin Closed Set Separation (MMCSS) Problem: *Given* non-empty subsets A, B of E, *find* disjoint closed sets $C_A, C_B \in \mathcal{C}_\rho$ with $A \subseteq C_A$, $B \subseteq C_B$ that maximize the margin, i.e., for all other disjoint closed sets $C'_A \supseteq A$, $C'_B \supseteq B$ it holds that $\mu_{C_A,C_B}(A, B) \geq \mu_{C'_A,C'_B}(A, B)$, if $\rho(A) \cap \rho(B) = \emptyset$; o/w return "No".

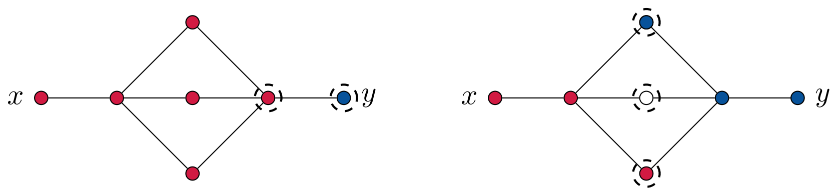


Figure 5.5: Example of a maximum margin half-space separation and a maximum margin closed set separation in a geodesic closure system over some graph. (left) A maximum margin half-space separation of x and y of margin 1 and (right) a maximum margin closed set separation of x and y of margin 2. The support elements are marked by dashed circles.

Remark 5.3.1. *The MMHSS problem is a special case of the MMCSS problem for $C_A = H, C_B = H^c$. Moreover, Lemma 5.3.2 implies that for any maximum margin closed set separation there exists a maximal closed set separation of the same margin. The converse is, however, not true in general. If fact, the maximum margin between two half-spaces solving the MMHSS problem can be smaller than the maximum margin between to maximal disjoint closed sets solving the MMCSS problem for the same input sets A, B, see Figure 5.5. The converse, i.e., that the maximum margin of a half-space separation is greater than that of a maximum margin closed set separation is not possible, because every maximum margin half-space separation is also a maximum margin closed set separation.*

A direct consequence of the negative result of Theorem 4.1.1 and Corollary 4.1.1 is the following:

Corollary 5.3.1. *The decision problem version of the MMHSS Problem, i.e., the question whether there exists a half-space separation of margin greater or equal k is* NP*-complete.*

Proof. The problem is in NP as we can easily check whether the output is a half-space separation and the margin has the proposed properties. Moreover we can reduce the problem to the Half-Space Separation problem (see Problem 4.1.1). □

In contrast to this negative results we can show that the MMCSS problem is solvable in polynomial time in the input with respect to the number of closure operator calls and linkage function evaluations. We solve the MMCSS problem by Algorithm 3, which is based on an adaptation of our greedy Algorithm 1. The input to the algorithm is an MLCS $(E, \mathcal{C}_\rho, l)$ together with two sets $A, B \subseteq E$ of training examples. We assume that $\mathcal{C}_\rho$ is given by the closure operator ρ, which returns the closure for any $X \subseteq E$ in unit time. Similarly, for any $X \subseteq E$ and $e \in E$, $l(X, e)$ is returned by another oracle in unit time. Accordingly, we measure the complexity of Algorithm 3 in terms of the number of closure operator calls and linkage function evaluations.

In Lines 1-3, the closures of A, B are calculated and checked for disjointness. In particular, if they are not disjoint, the algorithm terminates with "No", as in this case A and B are not separable by closed sets. Thus, the algorithm is correct for this case. Consider the case that $\rho(A) \cap \rho(B) = \emptyset$. For this case, all elements not contained in the

Algorithm 3: Maximum Margin Separation

```
   Input: a finite MLCS $(E, \mathcal{C}_\rho, l)$ and sets $A, B \subseteq E$
   Output: maximum margin closed sets $C_A, C_B \in \mathcal{C}_\rho$ with $A \subseteq C_A$ and $B \subseteq C_B$ if
           $\rho(A) \cap \rho(B) = \emptyset$; "No" otherwise
1  $\overline{A}, C_A \leftarrow \rho(A); \overline{B}, C_B \leftarrow \rho(B)$;
2  if $C_A \cap C_B \neq \emptyset$ then
3      return No;
4  $F \leftarrow E \setminus \{C_A \cup C_B\}$;
5  compute $\min\{l(\overline{A}, f), l(\overline{B}, f)\}$ for all $f \in F$ and sort $F$ by these values;
6  while $F \neq \emptyset$ do
7      take the smallest element $f \in F$;
8      if $(l(\overline{A}, f) \leq l(\overline{B}, f) \vee \rho(C_B \cup \{f\}) \cap C_A \neq \emptyset) \wedge \rho(C_A \cup \{f\}) \cap C_B = \emptyset$ then
9          $C_A \leftarrow \rho(C_A \cup \{f\})$;
10     else if $\rho(C_B \cup \{f\}) \cap C_A = \emptyset$ then
11         $C_B \leftarrow \rho(B_B \cup \{f\})$;
12     $F \leftarrow F \setminus (C_A \cup C_B \cup \{f\})$;
13 return $C_A, C_B$
```

union of the closures of A and B are first collected in F and sorted then by their minimum linkage from these two closed sets (Lines 4-5). The elements f in F will be processed one by one in this order and then immediately removed, potentially together with other untreated elements (Line 12). In particular, if the linkage from the closure of A to f is not greater than that of B or the current closed set C_B containing B cannot be extended by f, we expand the current closed set $C_A \supseteq A$ with f if it does not violate the disjointness with C_B (see Lines 8-9). Otherwise, we extend C_B by f, if $\rho(C_B \cup \{f\})$ remains disjoint with C_A (Lines 10-11). We then remove f and all other elements from F (Line 12) added to C_A or to C_B in Line 9 or 11.

An example of the algorithm to the case that $(E, \mathcal{C}_\rho, l)$ is defined over graphs with the geodesic or shortest path closure operator is given in Figure 5.6. We now show that Algorithm 3 is correct (Theorem 5.3.1) and efficient (Theorem 5.3.2). Furthermore, in case of Kakutani closure systems, the sets C_A, C_B returned in Line 13 form complementary half-spaces with maximum margin whenever $\rho(A) \cap \rho(B) \neq \emptyset$ (Corollary 5.3.2).

Theorem 5.3.1. *Algorithm 3 solves the MMCSS problem correctly.*

Proof. Let $(E, \mathcal{C}_\rho, l)$ be an MLCS and $A, B \subseteq E$. By construction, the algorithm returns "No" only for the case that $\rho(A) \cap \rho(B) \neq \emptyset$, i.e., when A and B are not separable in $\mathcal{C}_\rho$, implying the correctness for this case. Otherwise, the closed sets $C_A \supseteq A, C_B \supseteq B$ returned are disjoint and hence, form a *separation* of A and B. They are *maximal*, as only such elements of E are discarded that violate the disjointness condition. All such elements can be removed ultimately from F, as they do not have to be reconsidered again for the monotonicity of ρ.

Regarding optimality, suppose for contradiction that there are other disjoint closed sets $C'_A \supseteq A, C'_B \supseteq B$ such that

$$\mu_{C'_A,C'_B}(A,B) > \mu_{C_A,C_B}(A,B) \ . \tag{5.12}$$

For symmetry, we can assume without loss of generality that there is an $e^* \in C^c_A$ such that

$$\min\{l(\rho(A), C^c_A), l(\rho(B), C^c_B)\} = l(\rho(A), e^*) \ ,$$

i.e., $\mu_{C_A,C_B}(A,B) = l(\rho(A), e^*)$. Then, by (5.11) and (5.12) we have

$$l(\rho(A), e^*) < \min\{l(\rho(A), C'^c_A), l(\rho(B), C'^c_B)\} \tag{5.13}$$

implying $l(\rho(A), e^*) < l(\rho(A), C'^c_A)$. Thus, $e^* \notin C'^c_A$ and hence

$$e^* \in C'_A \subseteq C'^c_B \ . \tag{5.14}$$

But then, together with (5.13) and (5.14), we have

$$l(\rho(A), e^*) < l(\rho(B), C'^c_B) = \min_{x \in C'^c_B} l(\rho(B), x) \leq l(\rho(B), e^*) \ . \tag{5.15}$$

We prove that $e^* \in C'_A$ and $e^* \in C^c_A$ contradicts the assumptions. Conditions $C'_A \cap C'_B = \emptyset$ and $e^* \in C'_A$ imply that $\rho(\rho(A) \cup \{e^*\}) \cap \rho(B) = \emptyset$. Since $e^* \notin C_A$, e^* has not been added to C_A, though $l(\rho(A), e^*) < l(\rho(B), e^*)$ (5.15). But this can happen only if there is a non-empty set $G \subseteq F$ such that for all $g \in G$, g is before e^* in F, i.e., $\min\{l(\rho(A), g), l(\rho(B), g)\} \leq l(\rho(A), e^*)$. Assume there is a $g \in G$ such that $g \in C_A$, but $g \notin C'_A$. Then $g \in C'^c_A$ and thus,

$$\begin{aligned}
\mu_{C'_A,C'_B}(A,B) &= \min\{l(\rho(A), C'^c_A), l(\rho(B), C'^c_B)\} \\
&\leq \min\{l(\rho(A), g), l(\rho(B), g)\} \\
&\leq l(\rho(A), e^*) \\
&= \mu_{C_A,C_B}(A,B)
\end{aligned}$$

contradicting (5.12). Hence, for all $g \in G$, $g \in C_A$ implies $g \in C'_A$. In a similar way we have that $g \in C_B$ implies $g \in C'_B$ for all $g \in G$.

Since $e^* \in C^c_A$, $e^* \notin C_A$. There are two possible cases: (i) $e^* \in \rho(\rho(B) \cup G_B) \subseteq C'_B$, where $G_B \subseteq G$ is the set of elements added to $\rho(B)$. But this contradicts $e^* \in C'^c_B$. (ii) At the step e^* is considered for adding to C_A, there are disjoint subsets $G_A, G_B \subseteq G$ already added to $\rho(A)$ and $\rho(B)$, respectively, such that

$$\rho(\rho(\rho(A) \cup G_A) \cup \{e^*\}) \cap \rho(\rho(B) \cup G_B) \neq \emptyset.$$

But then, since $G_A \subseteq C'_A$ and $G_B \subseteq C'_B$ and by the monotonicity of ρ, we have $C'_A \cap C'_B \neq \emptyset$, as $e^* \in C'_A$; a contradiction. □

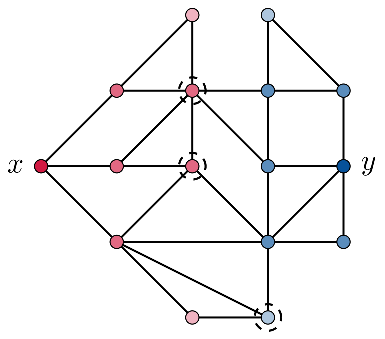


Figure 5.6: Maximum margin half-space separation of x and y defined by the shortest path closure. Brighter nodes are added later to the respective class. The maximum margin between $\{x\}$ and $\{y\}$ is 2 for the linkage defined by weight 1 for all edges (see Section 2.3). The support elements are marked by dashed circles.

Corollary 5.3.2. *For all MLCSs $(E, \mathcal{C}_\rho, l)$, Algorithm 3 solves the MMHSS-problem correctly if $(E, \mathcal{C}_\rho)$ is Kakutani.*

Proof. It is a direct implication of Theorem 5.3.1, as maximal disjoint closed sets are always half-spaces in any Kakutani closure system. □

Theorem 5.3.2. *Algorithm 3 requires at most $2 \cdot |E \setminus (\rho(A) \cup \rho(B))|$ evaluations of l and $2 \cdot |E \setminus (\rho(A) \cup \rho(B))| + 2$ calls of ρ.*

Proof. To sort F, we evaluate l twice for all $f \in F$ with $|F| = |E \setminus (\rho(A) \cup \rho(B))|$. The closure is calculated twice to determine the closures of the input sets (Line 1) and twice for all $f \in F$ in the worst case (Lines 8 and 10). □

5.4 Empirical Evaluations

In this section, we empirically evaluate the predictive performance of our methods developed in this chapter. For the evaluation we use synthetic and real-world data sets and consider binary classification tasks. We emphasize that our goal was to develop a very general separation algorithm that is *directly* applicable to entirely different domains. Hence, we do *not* compare its performance to domain-specific state-of-the-art methods, as they are usually based on domain-specific implementations incorporating domain-specific knowledge. Instead, we will compare our greedy algorithm developed in Section 5.1 to the maximum margin algorithm introduced in Section 5.3.2 that uses additional information in terms of proximities. Since we consider binary classification tasks, in case of finite point sets in $\mathbb{R}^d$ we compare our methods to support vector ma-

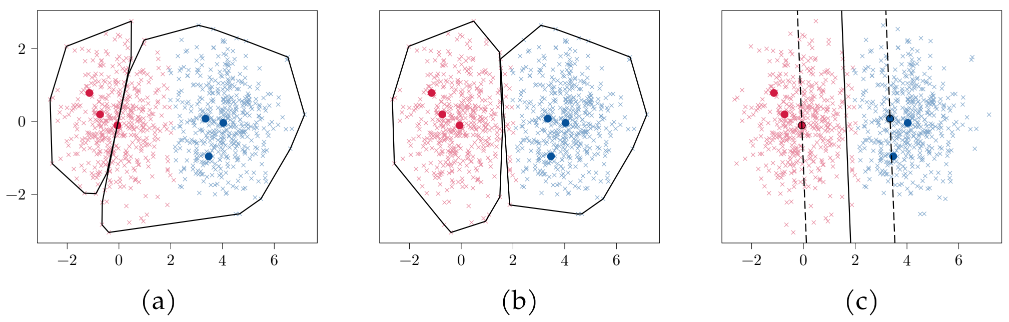


Figure 5.7: Comparison of greedy separation (a), maximum margin separation (b) and ordinary support vector machines (c).

chines. In case of graphs, we compare our results to the simple baseline that always predicts the majority class.

We recall the following supervised learning task considered: Let $(E, \mathcal{C})$ be a finite closure system. The ground set E is labeled according to an unknown binary valued function, i.e., then exist $L_1, L_2 \subseteq E$ with $L_1 \cap L_2 = \emptyset$ and $L_1 \cup L_2 = E$. For our synthetic datasets Synthetic2D, Synthetic3D, and Synthetic4D we know even more. By construction, the unknown target concept can be described by half-spaces, i.e., $L_1, L_2 \in \mathcal{C}$. For the real-world datasets, this latter condition does not necessarily hold, i.e., the labels do not form a half-space separation of the ground set. Given some training samples $T_1 \subseteq L_1$ and $T_2 \subseteq L_2$ such that $\rho(T_1) \cap \rho(T_2) = \emptyset$, the task is to predict the labels of the remaining elements in E. In particular, we run our algorithms using T_1 and T_2 as input sets. In Section 5.4.1 we first evaluate the performance of the maximal closed set and maximum margin separations on finite point sets in $\mathbb{R}^d$. For Algorithm 3, we chose the linkage function induced by the element-wise distances.

The second application described in Section 5.4.2 deals with the same supervised learning task, but on the domain of graphs. That is, the task is to classify the labels of vertices in trees and graphs of different sizes and edge densities. We compare the predictive performance of Algorithm 1 to that of Algorithm 3 and the simple baseline mentioned above. The only change we have to make is that the closure operator is now the geodesic closure over graphs. The monotone linkage function is defined by the pairwise distances of vertices in the graph (see Section 2.3). We evaluate our algorithms according to *accuracy* and *coverage* introduced in Section 2.6.

5.4.1 Binary Classification in Finite Point Sets

In this section, we consider point set separations in MLCSs over finite subsets of $\mathbb{R}^d$. The closure systems used in these experiments are given by the traces of convex hulls as defined in (2.1) on page 21; the linkage function by means of the Euclidean distance.

Experiments on Synthetic Data

We will now present the performance results of our algorithms on the synthetic datasets provided in Section 2.7.1. In addition to the quantitative results below, for one of the random datasets from $\mathbb{R}^2$, we visualize the output obtained by the three algorithms (see Figure 5.7). We selected three (in accordance with the VC-dimension of half-spaces in $\mathbb{R}^2$) random training examples for each class (denoted by dark blue resp. dark red). The class labels are indicated in light blue and light red. The predictions are given by the convex hulls for the two greedy algorithms and by the separating hyperplane for SVM.

We generated $1,000$ binary labeled variants of the Synthetic2D, Synthetic3D and Synthetic4D datasets as indicated in Section 2.7.1. Figure 5.8 shows the averaged accuracy (top row) and Figure 5.9 the averaged coverage (top row) for Algorithm 1, Algorithm 3, and the support vector machines for different training set sizes (see the values of the x-axes of Figure 5.8). The results obtained clearly show that maximum margin closed set separation outperforms the greedy separation algorithm in the predictive performance, especially on small training set sizes. Moreover, Algorithm 3 has a smaller standard deviation compared to Algorithm 1 as denoted by the colored areas around the mean curve. Furthermore, at least on the random datasets we used, it is also comparable to ordinary SVM by emphasizing that our definition is *not* a generalization of SVM; it is only an adaption of the idea of maximum margin separation to finite closure systems. The accuracy of the greedy algorithm strongly depends on the training set size and the dimension of the space, while the accuracy of the maximum margin algorithm is constantly above 0.9. Regarding the coverage, which measures how near the algorithm output is to some half-space separation, a similar behavior can be observed. Note that finite point sets in $\mathbb{R}^d$ are not half-space separable by MLCSs in general. While the average coverage for the greedy algorithm drops below 0.85 in case of $\mathbb{R}^4$ and 10 training samples, the maximum margin algorithm has an average coverage above 0.95 for all training set sizes. By definition, SVM always achieve a coverage of 1.

The empirical results reported in Figure 5.8 show that the predictive performance of maximum margin separation in this kind of closure systems is comparable to that of SVM and that it outperforms the greedy separation algorithm on finite synthetic point sets in $\mathbb{R}^2$, $\mathbb{R}^3$ and $\mathbb{R}^4$ that are half-space separable by construction.

Experiments on Real World Data

Regarding the binary classification experiments of the real-world datasets from Table 2.1, Figure 5.8 shows the average accuracy (bottom row) and Figure 5.9 the average coverage (bottom row) for the three algorithms for different training set sizes (see the values of the x-axes of Figure 5.8)[3]. Again, except for the Banana dataset, the results obtained show that maximum margin closed set separation outperforms the greedy separation algorithm in predictive performance. Indeed, for the Banana dataset, as it is highly not linear separable, support vector machines were also not able to find a good linear

[3]The average is taken over 100 different runs using different parts of the data as training samples.

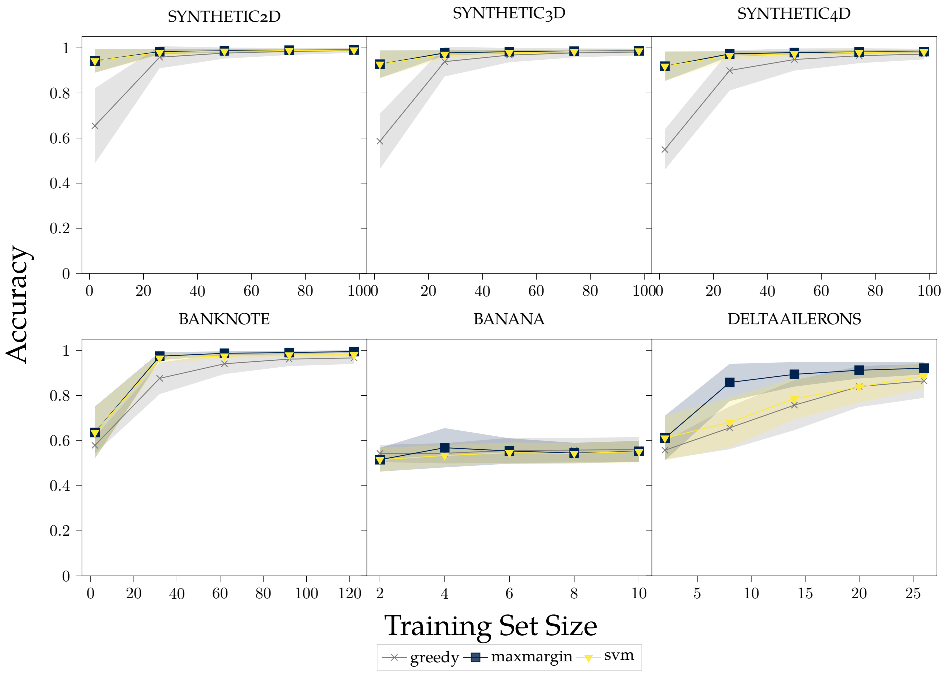


Figure 5.8: Accuracy of greedy separation, maximum margin separation, and SVM for point set classifications of the datasets presented in Section 2.7.1. The colored area denotes the standard deviation of the algorithms.

hypothesis. All the algorithms achieve an accuracy of around 0.55. It seems that also for real-world data that is mostly linearly separable, Algorithm 3 provides a predictive performance that is comparable or even better than ordinary support vector machines in terms of accuracy. In case of the Banknote dataset, the greedy algorithm achieves a maximum accuracy of 0.96, while the others achieve an accuracy of 0.98 if using more than 40 training samples. In case of DeltaAileron, the maximum margin algorithm achieves an accuracy of 0.92 compared to 0.89 obtained by support vector machines. This difference between the accuracies of Algorithm 3 and the support vector machine is a result of the comparison between two inherent problems as Algorithm 3 is not able to classify all elements whereas support vector machines assign a label to each element. Thus, we also measure the coverage of our algorithms in Figure 5.9.

Regarding the coverage, in Figure 5.9 it can be observed for all of the three real-world datasets that Algorithm 1 provides quite less coverage than the maximum margin separation Algorithm 3. This implies that not only the predictive performance of Algorithm 3 is better than Algorithm 1, but moreover, it finds solutions that are closer to some half-space separation. In particular, except for the DeltaAileron dataset, where Algorithm 1 respectively Algorithm 3 cover only 75% respectively 90% of the data on average, the output of Algorithm 3 is very near to a half-space separation.

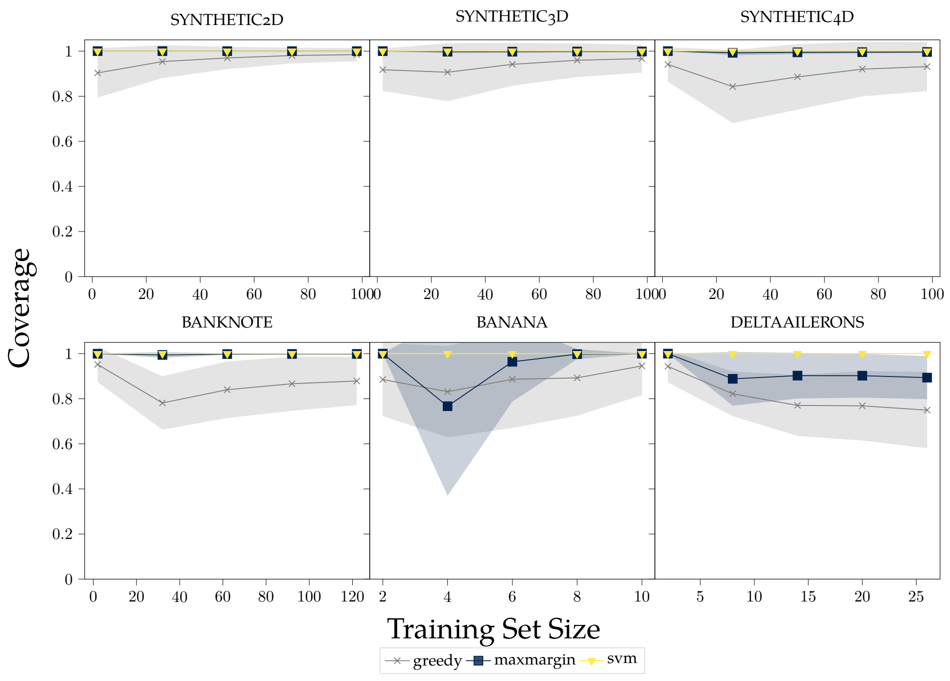
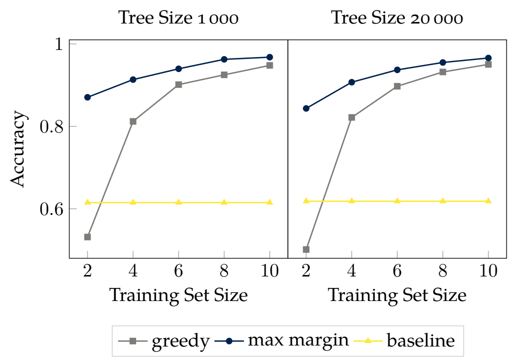


Figure 5.9: Coverage of greedy separation, maximum margin separation, and SVM for point set classifications of the datasets presented in Section 2.7.1. The colored area denotes the standard deviation of the algorithms.

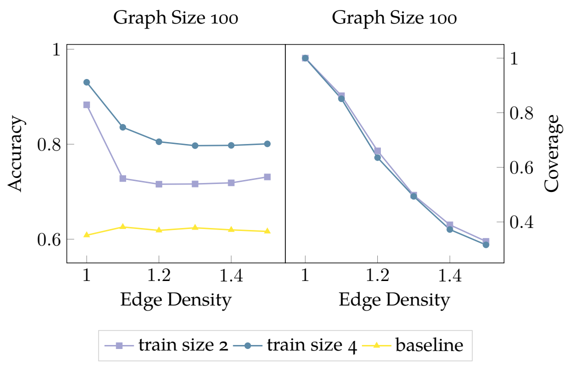
5.4.2 Vertex Classification in Random Graphs

For tree and graph data, we always consider the *geodesic closure* defined in (2.2), together with the monotone linkage function for (weighted) graphs as defined in Section 2.3. Nevertheless, we only consider unweighted graphs, i.e., we assume that each edge has a weight of one. In case of graphs, we are interested in binary node predictions of random connected graphs. Of course, the distribution of the labels in the graphs plays an important role in the prediction. Clearly, in case of randomly distributed labels, it is impossible to make any acceptable prediction by MLCSs defined by the closure operator in (2.2). Hence, we assume the following distributions of node labels in case of trees and graphs and analyze the predictive performance of our algorithm for different graph sizes and edge densities for the following two scenarios:

1. In case of trees, the nodes are labeled in a way that they form half-spaces, i.e., both label sets are closed and their union is the whole tree.
2. In case of graphs, we select two nodes at random and assign two different labels to them. Then the labels of the other nodes are determined by their distance to these center nodes. We ensure the subgraphs induced by the same class labels to be connected and randomly flip an unbiased coin to determine the label for nodes with the same distance.



(a) Comparison of greedy algorithm and maximum margin algorithm for node prediction task in random trees.



(b) Accuracy and Coverage of maximum margin separation in random graphs of different edge densities with 2 and 4 training samples.

Figure 5.10: Accuracy of vertex prediction in random trees and random graphs of different sizes and edge densities.

Moreover, in both cases we additionally use only graph labelings with nearly balanced class sizes, i.e., the minimum size of a class is at least 25% of the total size. In case of trees, we look at random trees of different sizes, ranging from $1,000$ to $20,000$ (see Fig. 5.10(a)). For each tree size and training sample size (see the x-axis of Fig. 5.10(a)), we generated $1,000$ binary labeled random trees in the above way. Then, for each run of the algorithm on a tree, $x/2$ training examples have been drawn at random from each of the two label sets for the input, where x is the x-axis value in Fig. 5.10(a). For evaluation, we run Algorithm 1 and Algorithm 3 on the training sets to predict the class labels of the unseen examples. The average accuracy, over all $1,000$ random trees is displayed in Fig. 5.10(a). As a baseline, we take the percentage defined by the majority class. Note that trees induce Kakutani closure systems and hence the coverage is always 1. One can see that with increasing training set size, the accuracy increases up to more than 0.95 in case of maximum margin separation and 10 training samples. Moreover, the maximum margin separation leads to better accuracy compared to the greedy separation, especially for small training sample sizes. Somewhat surprisingly, the tree size has no significant impact on the predictive performance.

In case of graphs with different edge densities[4], we generated $1,000$ random graphs for each edge density (see the x-axis values in Figure 5.10(b)) and assigned the nodes to one of the two classes as described above. The random graphs were generated from random trees by adding additional random edges until the required edge density has been reached. For each run of our algorithm, we selected 1 or 2 nodes from each label class at random for training such that their closures do not intersect. The accuracy results are shown in Fig. 5.10(b). We present also the coverage values, as the underlying MLCSs are not Kakutani in general. For increasing edge density, the accuracy decreases to 0.8 in case of 4 training samples and to 0.75 in case of 2 training samples for the edge density of 1.2. For edge density 1.5, there are no obvious changes in the accuracy. This can be explained by the fact that the coverage decreases to approximately 0.38 in case of an edge density of 1.5.

5.5 Summary and Open Questions

In this chapter, we have introduced the MCSS problem, a relaxation of the HSS problem. While the HSS problem requires an extension of two disjoint closed sets into a closed set partitioning of the *whole* space, the MCSS problem only requires an extension of disjoint closed sets into other disjoint closed sets that could not be extended further. In particular, solutions of the MCSS problem are *locally* maximal, while those to the HSS problem are *globally* maximal. That is, regarding the MCSS problem, there is no guarantee that *all* elements of the ground set are covered by one of the two disjoint closed sets. The goal of relaxing the original problem was to obtain a very general, but efficient algorithm that could be applied to *all* closure systems equipped with an efficiently computable closure operator. In this chapter, we achieved this goal by developing a simple greedy algorithm that solves the MCSS problem and calls the closure operator at most $2n + 2$-times, with n being the size of the ground set.

Although our algorithm is a simple greedy algorithm, we could show that in general, *no* other algorithm can perform better. As mentioned earlier, this worst case holds only for particular types of closure systems. Indeed, we have shown in Section 5.2 that a logarithmic improvement can be obtained for closure systems over lattices.

These results show the flexibility of our proposed greedy algorithm: One can easily integrate domain-specific knowledge for its improvements. This flexibility raises the question whether there are general strategies for adjusting the algorithm by using expert or domain-specific knowledge. Without going too much into the details, we sketch two possible options to integrate external knowledge. Recall that in Algorithm 1, we

(i) greedily choose an arbitrary unseen element (Line 5) and

[4]Given a graph $G = (V, E)$ by edge density we denote the value $\frac{|E(G)|-1}{|V(G)|}$. In particular, using this definition the density of trees is exactly one.

(ii) try to extend one of the two closed sets (say set A) with the element (Line 7).

(iii) If the extension was not possible, we try to extend the other set (say set B) (Line 9).

As we have seen for lattices, the number of closure operator calls does heavily rely on the particular choice of the next element in (i) and also on the extension order, i.e., the order of executing first (ii) and subsequently (iii) or vice versa. It is an interesting research problem to develop strategies optimizing the steps in (i),(ii), and (iii) for other domains (e.g., graphs) that improve our greedy algorithm.

In Section 5.3 we have presented another alternative approach based on integrating external knowledge to upgrade our original greedy algorithm. It follows the main idea of arriving at support vector machines (Boser et al., 1992) from Perceptrons (Rosenblatt, 1958). In fact, by replacing arbitrary separations with maximum margin separations, we have equipped closure systems with some similarity measure defined by monotone linkage functions. We again used the strength of the flexibility of our simple greedy algorithm and developed an extended version. This version provides for each disjoint closed input sets a maximal closed separation which is at the same time also a maximum margin separation.

Finally, we have evaluated our algorithms by considering concept learning over finite closure systems. That is, we have assumed that there is some unknown target concept that respects our hypothesis class, i.e., it is a half-space. Then, given the labels of some of the elements of the ground set, the task is to predict the labels of the remaining elements. Our empirical results in Section 5.4 show that for point classification in Euclidean spaces and for vertex classification in random graphs, the maximum margin algorithm clearly outperforms the greedy separation algorithm. These results clarify that it is possible to use the adaption of "linear" separation in finite closure system to solve classical machine learning tasks. Moreover, by incorporating further domain-specific knowledge by using monotone linkage functions, we can easily adjust our simply greedy algorithm to different domains.

Further potential applications of maximum margin closed set and half-space separation in finite closure systems include, among others, graphs, lattices (e.g., in inductive logic programming (Nienhuys-Cheng and de Wolf, 1997), formal concept analysis (Ganter et al., 2005), and itemset mining (Pasquier et al., 1999)), and finite point sets.

The next chapter is mainly devoted to the special case of geodesic closure systems over graphs, concentrating on computational aspects of the geodesic closure in graphs and providing another interesting application of the greedy Algorithm 1.

6 Practical Aspects of Mining and Learning in Finite Closure Systems

In this chapter we present two different practical aspects of the theory designed in the previous chapters. Our applications concentrate on the particular case of geodesic closure systems over graphs and their corresponding geodesic closure operator.

Approximating Geodesic Closed Sets Most of the time, we have treated the closure operator as a "black-box oracle", i.e., we did not care about the algorithmic realization and the runtime of the operator ρ. In practical applications, however, we also need to take into account the complexity of the specific closure operator considered. Obviously, it is impossible to analyze the complexity of every particular closure operator over all possible domains in a unified way. We therefore restrict our analysis to the *geodesic* closure operator in *graphs*. We motivate our particular choice by the fact that many promising works concerning mining and learning methods rely on geodesic closure systems over graphs (Cunha and Protti, 2019; de Araújo et al., 2019; Marc and Šubelj, 2018; Thiessen and Gärtner, 2020, 2021; Thiessen and Gärtner, 2022; Šubelj et al., 2019). For the reader's convenience, we recall the definition of geodesic closed sets in graphs (see (2.2) on page 21): Given a graph $G = (V, E)$ and a set $X \subseteq V(G)$, the *geodesic closure* of X is the *smallest* set $C \subseteq V(G)$ which contains X as well as *all* vertices on *all* shortest paths with both endpoints in C. Such a smallest set always exists, it is unique, and can be computed in $O(nm)$ time with a standard BFS algorithm (cf. Pelayo, 2013), where n (resp. m) denotes the number of vertices (resp. edges) of G. Thus, the total time complexity is $O(n^3)$, as $m = O(n^2)$ in the worst case. Accordingly, all approaches relying on computing geodesic convex hulls become practically *infeasible* for *large* networks.

Our first practical application is devoted to this runtime problem. One of our motivations is that in the analysis of the *geodesic core-periphery* decomposition of graphs proposed by Marc and Šubelj (2018), one has to calculate geodesic closed sets in large networks. For the cubic time complexity, Marc and Šubelj (2018) are able to study graphs only up to a size of around $5,000$ vertices while we consider graphs up to a size of around $5,000,000$ vertices, i.e., we look at graphs that are about $1,000$ times larger. It was observed by Marc and Šubelj (2018) that social networks primarily consist of a *dense* geodesic *core* surrounded by a *sparse periphery* (see Figure 2.7 for an example). While the geodesic core consists of those vertices that are contained in almost all geodesic closed sets of the closure system, the periphery can be described as the region of the graph where unique shortest paths are typically present.

To compute geodesic closed sets in large graphs, we give up the demand for *correctness* and propose a *novel* approach that calculates only an *approximation* of the closure of a vertex set $X \subseteq V(G)$. More precisely, we develop a heuristic that is based on the following main steps: Generate a set of *spanning subgraphs* of G independently at random, compute the closure of X in these subgraphs separately, and regard a vertex of G as an element of the geodesic closure of X if and only if it belongs to the closure of X in at least a user-specified percentage of the number of spanning subgraphs. The main question for this scheme is how to choose the class for the spanning subgraphs. We have three basic requirements regarding the spanning subgraphs. Firstly, sampling a single spanning subgraph needs to be at least *linear* in the number of edges of G. Secondly, calculating the closure in a sampled spanning subgraph needs to be faster than in G, and thirdly, the approximation quality needs to be reasonable. Regarding *forests*, a closer look at the problem and our empirical results reveal that already for graphs that are structurally very close to forests, only a poor approximation performance can be obtained in this way. This is because spanning forests may drastically distort the shortest paths.

Therefore, we consider the class of inclusion maximal *outerplanar graphs* (Chartrand and Harary, 1967) for spanning subgraphs because they fulfill the requirements above. Regarding the first step of the proposed heuristic, a maximal outerplanar spanning subgraph can be generated in $O(m)$ time (Djidjev, 2006). Since it was not possible to use this theoretical result in practice (see the discussion in Section 3 in the work of Leipert, 1998), we propose an alternative algorithm that is *linear* in m but returns only *almost* inclusion maximal outerplanar spanning subgraphs. For the second step, we present an algorithm computing the (geodesic) convex hull in an *outerplanar* graph G. The computation can be done in time $O(nf)$, where f, the *face number* of G, is the maximum number of *interior* faces over the biconnected components of G. Our algorithm is in fact *linear* in n in *practice* because f is typically *negligible* with respect to n. For example, in case of outerplanar spanning subgraphs of Erdős-Rényi random graphs with around one million edges, the average face number was consistently less than 80. Although the class of outerplanar graphs is only slightly beyond that of forests in the structural hierarchy, our empirical results with *large real-world* networks show that a close approximation of the geodesic convex hull can be obtained with outerplanar spanning subgraphs.

We empirically evaluate our heuristic on *large* real-world networks. The experiments clearly show that geodesic cores can be approximated closely (with a Jaccard similarity between 82 and 99%) with this scheme in *feasible* time, using only 100 spanning outerplanar subgraphs. Since we are unaware of any other approach approximating geodesic convex hulls in graphs, we compared the runtime results obtained by our heuristic with those of the standard algorithm mentioned above. In particular, in case of networks with more than 20 million (and up to 117 million) edges, the approximate decomposition could be computed in 5 hours or less with our algorithm. In contrast, the computation of the exact core-periphery decomposition with the standard algorithm had to be aborted after 50 *days*. Because of the close approximation, the approximate cores inherit several properties of the exact ones. For example as we show empirically, their degree distributions were consistently close to those of the exact ones.

Graph Tukey Depth The second practical application is concerned with the approximation of the *graph Tukey depth*. The *graph Tukey depth* is a relatively new centrality measure, introduced by Cerdeira and Silva (2021). Formally, the Tukey depth of a vertex v of a graph $G = (V, E)$ is defined by $\text{td}(v) := |V(G)| - |C|$ where C is a *largest* geodesic closed vertex set that does *not* contain v. The definition is based on the original notion of Tukey depth using the half-space separability of points in finite subsets of $\mathbb{R}^d$ (Tukey, 1975). Informally speaking, the semantics of Tukey depth in $\mathbb{R}^d$ is as follows: An element e of a finite ground set has *high* Tukey depth if it is "hard" to separate it from the rest of the set using separating hyperplanes only. Conversely, e has *low* Tukey depth if it is "easy" to separate it from the rest of the set. "Hard" respectively "easy" in this context mean that the half-space bounded by the separating hyperplane that contains e contains many, respectively a few other elements of the finite ground set. Thus, the elements' importance defined by ordinary Tukey depth in $\mathbb{R}^d$ relies on the possibility of separating them from other elements. Hence, since Tukey depth relies on set separations, it can be adapted to other domains using the (abstract) notion of half-space separation. In particular, this property of the original Tukey depth is utilized in the adaptation to *geodesic closure systems* over *graphs* (Cerdeira and Silva, 2021) and to arbitrary finite closure systems (see Section 2.5).

We use this definition to relate the concepts of graph Tukey depth and geodesic core-periphery decompositions (Marc and Šubelj, 2018). In particular, we experimentally show that the geodesic core of a graph consists of those vertices that are of *high* Tukey depth and that there is a gap in the depth distributions between the depths in the periphery and those in the core (see Figure 6.10). This observation allows for a new, *parameterized deterministic* definition of the cores, instead of the *probabilistic* definition given so far. That is, the *k-geodesic core* of a graph can be defined by the set of vertices with Tukey depth greater than a user-specified threshold k. Our empirical results clearly demonstrate that using the suitable threshold, the probabilistic definition of cores and observations in Marc and Šubelj (2018) can be interpreted and justified by our deterministic definition.

Similarly to the fact that the computation of the Tukey depth in $\mathbb{R}^d$ is NP-hard (Johnson and Preparata, 1978), it is also NP-hard to determine the graph Tukey depth of a vertex (Cerdeira and Silva, 2021). Motivated by this negative result and the observations given above, our main contribution is another *heuristic* algorithm for *approximating* the graph Tukey depth. It runs in time *polynomial* in the size of the input graph and approximates the Tukey depth of a vertex with a *one-sided* error by *overestimating* it. Our heuristic is based on Algorithm 1 that solves the more general *maximal closed set separation* (*MCSS*) problem. We are giving a very rough summary of the heuristic by recalling that Algorithm 1 is able to separate vertices from each other by constructing maximal closed disjoint sets. Hence, instead of computing a *global* maximum (i.e., by constructing a geodesic closed set of maximum size), we aggregate the information of several *local* maxima (i.e., sets that are maximal disjoint). Our experimental results with small graphs clearly demonstrate that the approximation is close to the exact Tukey depth, by noting that for larger graphs, we were not able to evaluate the approximation performance of our algorithm, as it was *not* possible to calculate the *exact* Tukey depth in practically feasible time.

Outline The rest of the chapter is organized as follows. In Section 6.1 we present our fast heuristic for approximating geodesic closures in graphs. Our results include the solution to the following two subproblems. First, we provide a linear time algorithm for sampling (almost) maximal outerplanar subgraphs (Section 6.1.1). Second, we develop a *fast* algorithm computing the geodesic closure for the particular class of outerplanar graphs. It does not depend on the input set size in contrast to the ordinary algorithm (Section 6.1.2). Finally, we empirically evaluate our two heuristics above as well as the application of the approximation heuristic on the task of approximating the geodesic core of large real-world networks (Section 6.1.3). In Section 6.2 we then analyze the connection between the graph Tukey depth and our results concerning half-space and maximal closed set separations. We first discuss potential applications of the graph Tukey depth to mining and learning with graphs (Section 6.2.1). Second, we present a heuristic approximating the graph Tukey depth (Section 6.2.2) and evaluate it on small graph datasets. We finish this chapter with some concluding remarks.

6.1 Approximating Geodesic Closures in Large Real-World Networks

In this section, we present a fast heuristic that approximates geodesic closed sets in large real-world graphs. We first give the details for the single steps of our heuristic and then empirically evaluate it by approximating geodesic cores in large real-world graphs. Let $G = (V, E)$ be some arbitrary graph and γ the corresponding geodesic closure operator. Our heuristic for computing the closure $\gamma(X)$ of a set $X \subseteq V(G)$ consists of the following three main steps:

(i) Generate s (inclusion) maximal outerplanar spanning subgraphs $G_1, \ldots, G_s$ of G independently at random, for some integer $s > 0$. Each (inclusion) maximal outerplanar spanning subgraph can be generated in $O(m)$ time, where $m = |E(G)|$ (Djidjev, 2006). In Section 6.1.1, we present a *fast* and *easy to implement* algorithm computing an *almost* inclusion *maximal* outerplanar spanning subgraph in $O(m)$ time. The number s of spanning subgraphs can be regarded as a constant (e.g., it was set to 100 in our experiments, independently of the networks' size). Thus, the total time of this step is *linear* in m in *practice*. We would like to add that this sampling can be considered as a preprocessing step as it has to be done only once for each graph.

(ii) For all outerplanar graphs G_i generated in step (i), calculate the closure $\gamma_{G_i}(X)$. Corollary 2.2.1 implies that $\gamma_{G_i}(X)$ can be computed in $O(n|X|)$ time. Below we give a more sophisticated algorithm. Its complexity is $O(nf)$, where f is the face number of G_i. Thus, its complexity is *independent* of the cardinality of X, which makes our algorithm *superior* to the standard one in terms of runtime. Since $f = O(n)$, it does not improve the *theoretical* worst-case complexity of the standard algorithm. Still, it has two crucial advantages over the standard algorithm. The first one is of *practical* interest: Our experiments with various graphs clearly show that the face number of spanning outerplanar graphs is *negligible*, compared to their size (i.e., number of vertices). The second one is of *theoretical* interest: Allowing only at most c faces per biconnected components in the spanning outerplanar graphs for some *constant* c, our closure algorithm runs in guaranteed *linear* time.

(iii) Finally, we define the approximate closure $\widetilde{\gamma_G}(X)$ as follows: A vertex $u \in V(G)$ is contained in $\widetilde{\gamma_G}(X)$ if and only if there is a set $S \subseteq \{G_1, \ldots, G_s\}$ with $|S| \geq t$ for some $0 < t \leq s$ integer such that $u \in \gamma_{G'}(X)$ for all $G' \in S$.

6.1.1 Generating Spanning Outerplanar Subgraph

Considering step (i) of the heuristic described above, the main contribution of this section is Algorithm 4, which generates a random spanning outerplanar subgraph for any undirected graph G in time *linear* in the order (i.e., number of edges) of G. According to Theorem 8 in (Djidjev, 2006), for any graph G one can generate a spanning *planar* subgraph in time linear in the order of G that is maximal with respect to planarity. Furthermore, the result of Djidjev (2006) can be modified in a way that it generates a *maximal* spanning *outerplanar* subgraph of G, also in linear time. However, to the best of our knowledge, no (simple) algorithmic realization exists of this result (cf. the discussion in Section 3.5 in the thesis of Leipert, 1998). Therefore, we propose an alternative algorithm that is *easy* to implement and *fast* in practice. Although we have no theoretical guarantee on the maximality of the spanning outerplanar subgraphs returned by our algorithm, our experimental results with Erdős-Rényi random graphs clearly show that they are *almost maximal* (i.e., only one or two edges are missing for maximality). Furthermore, for most graphs in our experiments, we obtained a relatively close approximation of their *geodesic cores* (Marc and Šubelj, 2018) already with almost maximal outerplanar graphs. Our algorithm can easily be modified so that the face number of the output spanning outerplanar graphs becomes *controllable* by some user-specified upper bound. In the particular case that the face number is bounded by some *constant*, our algorithm presented in Section 6.1.2 calculates the closure of any set of vertices in the output outerplanar graph in time *linear* in the number of vertices of the input graphs.

Similarly to (de Fraysseix and de Mendez, 2012), our sampling algorithm is based on utilizing a fundamental property of *Trémaux trees* of undirected graphs. More precisely, let G be an undirected graph and r some vertex of G. We assume, without loss of generality, that G is connected. Let T be a depth-first search (DFS) tree of G rooted at r. The connectivity of G implies $V(T) = V(G)$. We regard T as a *sorted* tree, where the DFS traversal of G defines the order of the children of the vertices. It is a well-known fact that T is a Trémaux tree of G, i.e., for all *back edges* $\{v, w\} \in E(G) \setminus E(T)$ we have $v \preccurlyeq w$ or $w \preccurlyeq v$, where for all $x, y \in V(T)$, $x \preccurlyeq y$ if and only if Path(r, y) in T contains x. In what follows, for all $v, w \in V(T)$, $p(v)$ denotes the parent of v in T, $d(v)$ stands for the *depth* of v (i.e., the length of Path(r, v) in T), and a back edge $\{v, w\} \in E(G) \setminus E(T)$ with $w \preccurlyeq v$ is denoted by (v, w).

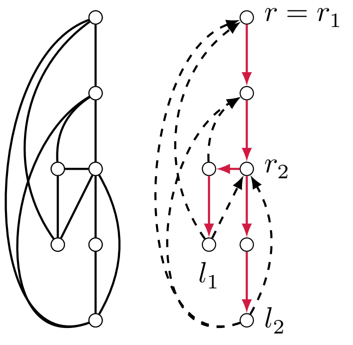
It holds that the DFS traversal of G defines a sequence of paths denoted by $P_1 = \text{Path}(r_1, l_1), \ldots, P_k = \text{Path}(r_k, l_k)$ in T, where $r_1 = r$, l_1 is the leftmost leaf of T, r_{i+1} is the deepest vertex of $\text{Path}(r, l_i)$ such that it has a child not belonging to $\bigcup_{\ell \leq i} V(P_\ell)$, and l_{i+1} is the leftmost leaf in the subtree of T rooted at r_{i+1} that is not an element of $\bigcup_{\ell \leq i} V(P_\ell)$ $(1 \leq i < k)$. For all back edges (v, w) with $v \in V(P_i)$ it holds that w is a vertex of $\text{Path}(r, l_i)$. The sequence $P_1, \ldots, P_k$ is referred to as the (ordered) sequence of *DFS paths* of G with respect to T.

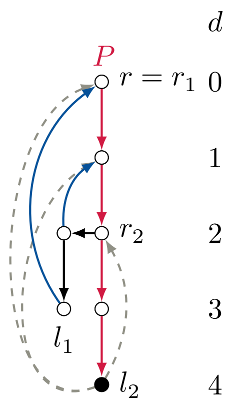
Let G' be a connected outerplanar graph, T a DFS tree of G' rooted at r for some $r \in V(G')$, and let $\text{Path}(r_1, l_1), \ldots, \text{Path}(r_k, l_k)$ be the sequence of DFS paths of G' with respect to T. Then G' has an embedding in the plane with the following properties: For all leafs l of T and for all vertices u on $P = \text{Path}(r, l)$, the images of the vertices v of G' in $\mathbb{R}^2$ can be rotated around that of r, all in the same direction as u, such that the x-coordinate of the point representing u becomes equal to that of r, the new embedding preserves the non-crossing edge property, and all back edges with both endpoints in P are either on the left- or on the right-hand side of the vertical line containing P. Notice that if a back edge belongs to more than one path from a leaf to the root, then it is either to the left or right for *all* such paths. The set of left (resp. right) back edges of G' is denoted by $\mathcal{L}$ (resp. $\mathcal{R}$). A vertex x of G' lying on $\text{Path}(r, l_i)$ is *reachable* from *left* (resp. *right*) with respect to $\text{Path}(r_i, l_i)$ if there is no left (resp. right) back edge (v, w) with $w \prec x \prec v$ such that $x \prec r_i$ or $v \preccurlyeq l_i$.

Let G be a connected graph, T a DFS tree of G, and G' be a spanning outerplanar subgraph of G containing T as a subgraph. We assume that G' is embedded into the plane as sketched above, i.e., all back edges of G' are either left or right back edges. A back edge $(v, w) \in E(G) \setminus E(G')$ is *valid* if it can be added to G' as a *left* or *right* back edge such that it intersects no other edges from G' and for all vertices, v in the resulting graph there exists a path P_i of T such that v is reachable from left or right with respect to P_i. One of the crucial steps in generating a spanning outerplanar graph of G with respect to T is to check the validity of the back edges. We introduce some additional notions to decide this problem in *constant* time. More precisely, let $P_i = \text{Path}(r_i, l_i)$ be a DFS path of T and v be a vertex with $r_i \preccurlyeq v$. Then

- $\text{reach}(v, P_i) \subseteq \{R, L\}$ denotes the direction(s) from which v can be reached in G' with respect to P_i,
- $\uparrow_L(v)$ (resp. $\uparrow_R(v)$) denotes the smallest depth of the vertex w on $\text{Path}(r, v)$ in T such that (v, w) is a valid left (resp. right) back edge, and
- $\sigma_L(v)$ (resp. $\sigma_R(v)$) is True if there are $P = \text{Path}(r, l_j)$ and $u, w \in V(P)$ for some j $(1 \leq j \leq k)$ such that $v \in V(P)$, (u, w) is a left (resp. right) back edge, and $w \prec v \prec u$; o/w it is False.

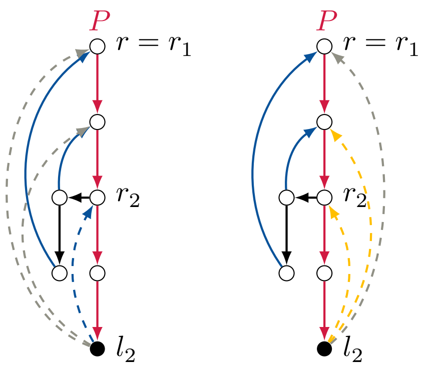
Using the above notions and notation, we are ready to present Algorithm 4. In Lines 1–2 it first computes a DFS tree of the input graph G for some arbitrary root $r \in V(G)$. In Lines 3–5 it initializes some variables. In particular, the left (resp. right) valid back edges that will be added to T will be stored in the set variables $\mathcal{L}_l$ (resp. $\mathcal{R}_l$). Since no





(a) *Left:* Input graph G, *Right:* DFS-Traversal of G rooted at r with DFS edges in red and back edges in dashed black. The paths are $P_1 = \text{Path}(r_1, l_1)$ and $P_2 = \text{Path}(r_2, l_2)$.

(b) *Blue Edges:* Already added left edges for path $\text{Path}(r_1, l_1)$. *Gray Edges:* Still unprocessed edges in the current path $P = \text{Path}(r, l_2)$ drawn as a vertical line. d denotes the node depth in P. The blue arrows denote valid back edges in $\mathcal{L}$.



reach	σ_L	σ_R	$\uparrow_L$	$\uparrow_R$
$\{L, R\}$	F	F	0	0
$\{R\}$	T	F	0	0
$\{L, R\} \rightarrow \{L\}$	T	$F \rightarrow T$	2	1
$\{L, R\} \rightarrow \{L\}$	F	$F \rightarrow T$	2	1
$\{L, R\}$	F	F	$2 \rightarrow 3$	1

(c) Running the subroutine AddEdges for l_2. *Left:* Determine E_L (valid left back edges in dashed blue), *Right:* Determine E_R (valid right back edges in dashed orange). The table shows the algorithm parameters before adding the orange edges (black numbers); changes after adding the orange edges to $\mathcal{R}$ are marked in orange.

Figure 6.1: (6.1(a)) shows an example of a DFS-Traversal of a graph G, (6.1(b)) shows some possible intermediate result of Algorithm 4 with unconsidered back edges starting at l_2 in dashed grey. (6.1(c)) shows the results of the AddEdges subroutine with edges in E_L marked in dashed blue and edges in E_R marked in dashed orange.

Algorithm 4: Spanning Outerplanar Subgraph

```
   Input: connected graph $G$
   Output: spanning outerplanar subgraph $H$ of $G$
1  select a vertex $r \in V(G)$ at random;
2  generate a DFS tree $T$ of $G$ rooted at $r$ and with DFS paths
     $P_i = [v_1 = r_i, \ldots, v_{n_i} = l_i]$ $(1 \le i \le k)$;
3  $\mathcal{L}_0, \mathcal{R}_0 \leftarrow \emptyset, l \leftarrow 0$;
4  $\uparrow_L(r), \uparrow_R(r) \leftarrow 0$;
5  $\sigma_L(v), \sigma_R(v) \leftarrow$ False for all $v \in V(T)$;
6  for $i = 1, \ldots, k$ do
7      reach$(r_i) \leftarrow \{R, L\}$;
8      for $\delta \in \{R, L\}$ do
9          if $\sigma_\delta(r_i) \vee r_i = r$ then $\uparrow(r_i) = d(r_i)$;
10         else $\uparrow_\delta(r_i) = \uparrow_\delta(p(r_i))$ ;
11     for $j = 2, \ldots, n_i$ do
12         reach$(v_j) \leftarrow \{R, L\}$;
13         $\uparrow_L(v_j) = \uparrow_L(p(v_j)), \uparrow_R(v_j) = \uparrow_R(p(v_j))$;
14         $F = \{(v_j, w) \in E : w \prec v_j\}$;
15         $(E_L, E_R) =$ AddEdges$(v_j, F)$;
16         $l \leftarrow l + 1$;
17         $\mathcal{L}_l = \mathcal{L}_{l-1} \cup E_L, \mathcal{R}_l = \mathcal{R}_{l-1} \cup E_R$;
18 return $H = (V, E(T) \cup \mathcal{L}_l \cup \mathcal{R}_l)$;
```

back edge going out from the root can be added to T, $\uparrow_L(r)$ and $\uparrow_R(r)$ are both set to 0. Furthermore, the Boolean variables $\sigma_L(v), \sigma_R(v)$ are set to False for all $v \in V$, as T has no back edge initially.

The algorithm then processes the DFS paths $P_1, P_2, \ldots, P_k$ of T in their DFS order defined above (cf. loop 6–17). For each $P_i = \text{Path}(r_i, l_i)$, it adds greedily as many as possible back edges to Path(r, l_i) with at least one endpoint in P_i such that the extension does not violate outerplanarity. In particular, it considers the vertices of P_i one by one, from r_i towards l_i. While processing the vertices of P_i, their reachability is set to $\{L, R\}$ (cf. Lines 7 and 12). Since we have not yet added any back edge to P_i, all of them are reachable from the left and right with respect to P_i. For simplicity, we omit the reference path Path(r, l_i) from the notation by noting that all vertices above r_i inherit their reachability state with respect to Path(r, l_{i-1}).

For all vertices v of P_i, $\uparrow_L(v)$ (resp. $\uparrow_R(v)$) is set to the depth of r_i (cf. Line 9) if (i) $v = r_i$ and there is a j, $1 \le j < i$, such that r_i is *not* reachable from left (resp. right) with respect to Path(r, l_j) or (ii) $v = r$; o/w to $\uparrow_L(p(v))$ (resp. $\uparrow_R(p(v))$) (cf. Lines 10 and 13). Regarding the first case, there is no valid left/right back edge going out from v satisfying (i) or (ii), and hence, its left/right smallest depth cannot be smaller than $d(v)$. For all other cases, if a back edge (v, w) added to left (resp. right) destroys the reachability of

Algorithm 5: Function AddEdges

```
Assumed: undirected graph $G$ and DFS tree $T$ of $G$
Input: $v \in V(G)$ and a set $F$ of back edges, all with initial vertex $v$
Output: $E_L, E_R \subseteq F$ with $E_L = \emptyset$ or $E_R = \emptyset$
1  $E_L, E_R \leftarrow \emptyset$;
2  for $\delta \in \{L, R\}$ do
3      for $(v, w) \in F$ do
4          if $\delta \in \mathrm{reach}(w)$ and $\uparrow_\delta(v) \leq d(w)$ then
5              add $(v, w)$ to $E_\delta$;
6  $X \leftarrow L, Y \leftarrow R$;
7  if $|E_R| > |E_L|$ then $X \leftarrow R, Y \leftarrow L$;
8  if $E_X \neq \emptyset$ then
9      $\uparrow_Y(v) = d(p(v))$;
10     for $(v, w) \in E_X$ do
11         for $x$ in the open intervall $(v, w)$ do
12             delete $X$ from reach$(x)$;
13             $\uparrow_Y(x) = d(x)$;
14             $\sigma_X(x) = $ True;
15 if $X = L$ then return $(E_L, \emptyset)$;
16 else return $(\emptyset, E_R)$;
```

some vertex x with $w \preccurlyeq x$, then all other back edges (v', w') with $v \preccurlyeq v'$ and $w' \preccurlyeq w$ added to left (resp. right) also destroy it. Hence, it suffices to save the vertex with the lowest depth, which is a valid endpoint for a back edge added to the left (resp. right). After all relevant pieces of information have been calculated for v_j, we take the set of all possible back edges from v_j ending in some vertex $w \prec v_j$ (Line 14) and compute a maximal subset of this set of edges in function AddEdges that can be added to Path(r, l_i) without destroying outerplanarity (Line 15).

Function AddEdges is specified in Algorithm 5. Its input consists of a vertex v of T and a set F of candidate back edges for Path(r, l_i) processed currently by Algorithm 4, each with starting vertex v. Algorithm 5 tries to add as many as possible edges of F to Path(r, l_i), either all from left or from right, without violating outerplanarity. In particular, each back edge (v, w) is checked in loop 2–5 for left and right validity with respect to Path(r, l_i) (cf. the condition in Line 4) and, depending on the outcome of the test, is added to E_L and E_R. As an example, all the gray edges in Fig. 6.1(c) violate at least one of the two conditions in Line 4 of Algorithm 5, while the colored edges fulfill both of them. Notice that once a back edge $(v, w) \in F$ has been added to one of the sides of Path(r, l_i), then *no* back edge $(v, w') \in F$ can be added to its other side, as reach$(p(v)$, Path$(r, l_i))$ became empty, violating the reachability property of $p(v)$. Thus, we can add either all edges from E_L to the left or all edges from E_R to the right side of Path(r, l_i). Since our goal is to maximize the number of back edges in G', we take the set with the greater cardinality (cf. Lines 6–7).

After the selection of one of the two sets, say E_L (the case of E_R is analogous), we update the reachability information of the vertices on $\text{Path}(r, v)$ as follows: Since all back edges are of length at least 2, no back edge (s, t) with $l_i \succ s \succ v \succ p(v) \succ t$ can be added to the right of $\text{Path}(r, l_i)$, as $p(v)$ became unreachable from both directions. Therefore, $\uparrow_R(v)$ has to be set to $d(p(v))$ (cf. Line 9). Furthermore, for all back edges $(v, w) \in E_L$ and for all internal vertices x of $\text{Path}(v, w)$, x becomes unreachable from left (i.e., L must be deleted from the reachability set of x with respect to $\text{Path}(r, l_i)$). Moreover, (v, w) prohibits any left back edge in any possible path $P_j = \text{Path}(r_j, l_j)$ with $x = r_j$ (i.e., $\sigma_L(x)$ has to be set to True) (cf. Line 14) and the terminal vertex of any right back edge in P_j cannot be smaller concerning the depth than $d(p(x))$ (i.e., $\uparrow_R(x)$ has to be set to $d(x)$). In our example in Figure 6.1(c), by adding the orange edges to $\mathcal{R}$ we update the parameters to the orange values (see, also, the table in Figure 6.1(c)).

We are ready to state the main result of this section.

Theorem 6.1.1. *For any connected graph G, Algorithm 4 returns a spanning outerplanar subgraph of G in $O(|E(G)|)$ time.*

Proof. The proof follows directly from Lemma 6.1.2 and Lemma 6.1.3. □

In the proof of Lemma 6.1.2 concerning the correctness of Algorithm 4, we will use the following auxiliary lemma.

Lemma 6.1.1. *The set $\mathcal{L}_l$ (resp. $\mathcal{R}_l$) of back edges computed in Algorithm 4 fulfills the following properties for all $l \geq 0$:*

(*i*) *For all $(v_1, w_1), (v_2, w_2)$ in $\mathcal{L}_l$ (resp. $\mathcal{R}_l$) and $y \in V(G)$ satisfying $w_2 \prec y \preccurlyeq v_1$ and $w_2 \prec y \preccurlyeq v_2$,*

$$w_1 \prec w_2 \implies v_2 \preccurlyeq v_1 \tag{6.1}$$
$$w_1 = w_2 \implies v_1 \prec v_2 \text{ or } v_2 \prec v_1 \ . \tag{6.2}$$

(*ii*) *For all $(v_a, w_a) \in \mathcal{L}_l$ (resp. $\mathcal{R}_l$) and $x \in V(T)$ with $w_a \prec x \prec v_a$, there is no $(v_b, w_b) \in \mathcal{R}_l$ (resp. $\mathcal{L}_l$) with $v_a \preccurlyeq v_b$ and $w_b \preccurlyeq w_a$.*

Proof. We prove both claims with induction on l for the direction left; the proof of the other direction is analogous. Regarding (i), the proof of the base case $l = 0$ is trivial. Suppose (i) holds for $l \geq 0$ and let $(v_1, w_1), (v_2, w_2) \in \mathcal{L}_{l+1}$. (Case 1) If $(v_1, w_1), (v_2, w_2) \in \mathcal{L}_l$, then (i) holds by the induction hypothesis. (Case 2) If $(v_1, w_1), (v_2, w_2) \in \mathcal{L}_{l+1} \setminus \mathcal{L}_l$, then $v_1 = v_2$ and $w_1 \neq w_2$. Hence, (6.1) and (6.2) both hold for this case. (Case 3) If $(v_1, w_1) \in \mathcal{L}_{l+1} \setminus \mathcal{L}_l$ and $(v_2, w_2) \in \mathcal{L}_l$, then the order of processing the vertices of T implies

$$v_1 \not\prec v_2 \ . \tag{6.3}$$

Moreover, as (v_1, w_1) is a left back edge, we have

$$\uparrow_L(v_1) \leq d(w_1) \tag{6.4}$$

(cf. the condition in Line 4 of Algorithm 5 for $\delta = L$). Suppose

$$w_1 \prec w_2 \ . \tag{6.5}$$

Then $w_1 \prec w_2 \prec v_1$. Assume for contradiction that v_1 and v_2 are incomparable. Then they lie on different paths in T, implying $w_2 \prec y \prec v_2$ for y in (i). Since, by condition of this case, (v_2, w_2) has been added to T before (v_1, w_1), v_2 was considered before v_1 in the DFS traversal. Hence, there exists $r_i \prec v_1$ such that

$$w_2 \prec r_i \prec v_2 \ . \tag{6.6}$$

Thus, after (v_2, w_2) has been added to $\mathcal{L}_j$ for some $j \leq l$, we certainly have $L \notin \text{reach}(r_i)$ and $\sigma_L(r_i) = \text{True}$ (cf. Lines 12 and 14 in Algorithm 5). In a later step, when processing v_1, we therefore have

$$\uparrow_L(v_1) \geq d(r_i) \tag{6.7}$$

(cf. Lines 9 and 10 of Algorithm 4). By (6.5) and (6.6) we have $w_1 \prec w_2 \prec r_i$, from which $d(w_1) < d(w_2) < d(r_i) \leq \uparrow_L(v_1)$ follows by (6.7). However, this contradicts (6.4). Thus, v_1 and v_2 are comparable and hence, together with (6.3), implication (6.1) holds for Case 3. The proof of (6.1) for the last case (Case 4) that $(v_1, w_1) \in \mathcal{L}_l$ and $(v_2, w_2) \in \mathcal{L}_{l+1} \setminus \mathcal{L}_l$ is analogous.

Implication (6.2) can be shown with similar arguments, so it remains to prove claim (ii) of the lemma. The base case $l = 0$ holds trivially. For the induction step, let $(v_a, w_a) \in \mathcal{L}_{l+1}$ with $w_a \prec x \prec v_a$. Assume first $(v_a, w_a) \in \mathcal{L}_l$ and suppose for contradiction that there is a $(v_b, w_b) \in \mathcal{R}_{l+1}$ with $v_a \preccurlyeq v_b$ and $w_b \preccurlyeq w_a$. By the induction hypothesis, it must be the case that $(v_b, w_b) \in \mathcal{R}_{l+1} \setminus \mathcal{R}_l$. Then

$$\uparrow_R(v_b) \geq \uparrow_R(v_a) \geq d(p(v_a)) \geq d(x) > d(w_b) \ ,$$

where $\uparrow_R(v_a) \geq d(p(v_a))$ holds by Line 9 of Algorithm 5. Hence (v_b, w_b) does *not* satisfy the condition in Line 4 in Algorithm 5 for $\delta = R$, contradicting $(v_b, w_b) \in \mathcal{R}_{l+1}$. A contradiction for the case that $(v_a, w_a) \in \mathcal{L}_{l+1} \setminus \mathcal{L}_l$ and $(v_b, w_b) \in \mathcal{R}_l$ can be obtained similarly by noting that we cannot have $(v_a, w_a) \in \mathcal{L}_{l+1} \setminus \mathcal{L}_l$ and $(v_b, w_b) \in \mathcal{R}_{l+1} \setminus \mathcal{R}_l$ (cf. Lines 15 and 16 of Algorithm 5). □

Lemma 6.1.2. *For all $l \geq 0$, $G_l = (V(G), E(T) \cup \mathcal{L}_l \cup \mathcal{R}_l)$ is outerplanar after iteration l of loop 11–17 of Algorithm 4.*

Proof. We show by induction on l that G_l can be drawn in the plane in a way that

(i) all edges $(v, w) \in \mathcal{L}_l$ (resp. $\mathcal{R}_l$) that have been added to the DFS path P_i for some i $(1 \leq i \leq k)$ lie left (resp. right) with respect to P_i and do not intersect any other edge of G_l,

(ii) all vertices of G_l lie on the outer face.

The proof of the case $l = 0$ is automatic. For the induction step, suppose G_l has an embedding in the plane that satisfies (i)–(ii) and consider the sets $\mathcal{L}_{l+1}^{\text{new}} = \mathcal{L}_{l+1} \setminus \mathcal{L}_l$ and $\mathcal{R}_{l+1}^{\text{new}} = \mathcal{R}_{l+1} \setminus \mathcal{R}_l$. If both of them are empty, then the claim holds by the induction hypothesis. Otherwise, exactly one of them, say $\mathcal{L}_{l+1}^{\text{new}}$, is non-empty by Lines 15 and 16 of Algorithm 5; the proof of the case $\mathcal{R}_{l+1}^{\text{new}} \neq \emptyset$ is analogous. Let $P_i = \text{Path}(r_i, l_i)$ be the DFS path $(1 \leq i \leq k)$ and v be a vertex of P_i such that the left back edges in $\mathcal{L}_{l+1}^{\text{new}}$ have been constructed for v in the outer loop of Algorithm 4. Then $v \neq r_i$ and it is the initial vertex of all back edges in $\mathcal{L}_{l+1}^{\text{new}}$. Thus, each edge in $\mathcal{L}_{l+1}^{\text{new}}$ can be drawn left with respect to P_i, without intersecting any other edge in $\mathcal{L}_{l+1}^{\text{new}}$. All edges in $\mathcal{R}_l$ with an endpoint in $\text{Path}(r, l_i)$ are right with respect to P_i. Hence, the edges in $\mathcal{L}_{l+1}^{\text{new}}$ can be drawn without intersecting these right back edges. Suppose for contradiction that there is a new edge $(v_2, w_2) \in \mathcal{L}_{l+1}^{\text{new}}$ that cannot be drawn left with respect to P_i without crossing some other edge $(v_1, w_1) \in \mathcal{L}_l$. By the induction hypothesis, (v_1, w_1) could be drawn for some $l' \leq l$ iteration also left with respect to P_i, without crossing any other edge. Hence, it must be the case that $w_1 \prec w_2$, $w_2 \prec y \preccurlyeq v_1$, and $w_2 \prec y \preccurlyeq v_2$, where y is the vertex with the largest depth satisfying $y \preccurlyeq v_1, v_2$. But then, $v_2 \preccurlyeq v_1$ by Lemma 6.1.1, contradicting that (v_1, w_1) has been considered before (v_2, w_2). Thus, (v_2, w_2) can be drawn left with respect to P_i without intersecting any edges in $\mathcal{L}_l$, completing the proof of (i).

To prove (ii), notice that if $(v, w) \in \mathcal{L}_{l+1}$ destroys the outerplanarity of the graph, then $(v, w') \in \mathcal{L}_{l+1}$ with $w' \prec w$ does the same. Thus, it suffices to consider the back edge in $\mathcal{L}_{l+1}$ with the terminal vertex of the smallest depth. Let (v, w^*) be this back edge. We show that it is possible to add (v, w^*) to the planar embedding so that all vertices x lie on the outer face and (i) stays valid. This is straightforward by induction for all vertices $x \in V(T)$ with $w^* \not\prec x$, so it suffices to consider the remaining sets $V_1 = V(\text{Path}(w^*, v))$ and $V_2 = \{x \in V(T) \setminus V_1 : w^* \prec x\}$.

We first prove the claim for the vertices in V_1. Suppose for contradiction that there is a vertex $x \in V_1$ that does not lie on the outer face. This can happen if and only if there is $(v_R, w_R) \in \mathcal{R}_l$ such that $w^* \prec x \prec v$ and $w_R \prec x \prec v_R$. However, this contradicts (ii) of Lemma 6.1.1. Regarding the other case, assume there is an $x \in V_2$ that does not lie on the outer face. Let $x^* \in V_1$ be the vertex with maximum depth such that $x^* \prec x, v$. Assume there is an edge $(v', w') \in \mathcal{L}_l$ with $w' \prec x^* \prec v'$. Since one of $w' \prec w^*$, $w^* \prec w'$, and $w' = w^*$ holds, the condition of (i) of Lemma 6.1.1 is fulfilled for $x^* = y$. However, v', v are incomparable, implying that such an edge does not exist. Hence, we can redraw the outerplanar subgraph consisting of all vertices y with $y \succcurlyeq x^*$ right to P_i such that all of its vertices and all other vertices of the graph lie on the outer face. Moreover, it can be redrawn such that no edges are crossing and the redrawing fulfills (i) because no back edge in the subgraph of all vertices y with $y \succcurlyeq x^*$ lies left with respect to P_i, completing the proof of (ii). □

Lemma 6.1.3. *For any connected graph G, Algorithm 4 terminates in $O(|E(G)|)$ time.*

Proof. Note first that T in Line 2 can be computed in $O(|E|)$ time. The condition in Line 9 is checked $2l$-times, where $l = O(|V(G))$ is the number of leafs of T. Since each vertex v of T is considered at most once in the second inner loop (cf. Line 11) and the number of

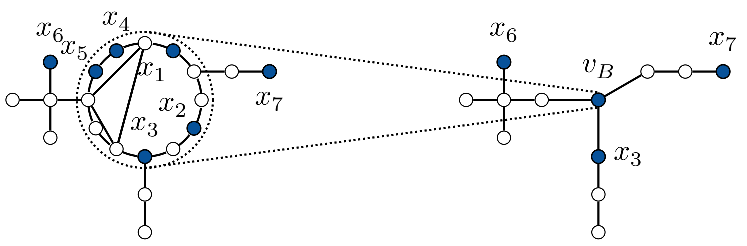


Figure 6.2: *Left:* Outerplanar graph G and input set $X = X_0 = \{x_1, \ldots, x_7\} \subseteq V(G)$ in blue, *Right:* BB-tree $\tilde{G}$ constructed from G (for the biconnected outerplanar component on the left-hand side, see the dotted circle, a new node v_B is added). The sets $C_1 = \{v_B\}$ and $C_2 = \{x_3, x_6, x_7\}$ are marked in blue (see Lines 1–5 of Line 1).

edges added to F in Line 15 is bounded by the degree of v, the total time for this inner loop is $O(|E(G)|)$.

Finally, AddEdges is called at most $|V(G)|$ times in Line 15. It can be checked in constant time whether an edge in F can be added to E_L or E_R. If a back edge (v, w) can be added, we have to update the properties of vertices between (v, w) (Line 10). This can be done with a *naive* algorithm in a quadratic runtime. However, we can store the vertices that are reachable from both *left* and *right* during the iteration over all P_i in a global stack. If an edge (v, w) is added, we remove all vertices starting with the parent of v from the stack unless we have found w (w will not be deleted from the stack); this is because these vertices cannot be the endpoint of other *left* or *right* edges. The runtime of this operation is linear in the number of elements removed from the stack. Once a vertex has been removed from the stack, it will never be added again to it, except for the case that it is equal to r_i for some i $(1 \leq i \leq k)$. Hence the *overall* runtime of this operation is at most linear in the number of edges, implying the claimed total runtime of $O(|E(G)|)$. □

6.1.2 Geodesic Closure in Outerplanar Graphs

The previous section shows that it is possible to sample almost maximal spanning outerplanar graphs in time linear in the number of edges of the ground graph. To achieve our goal of developing a fast heuristic for approximating geodesic closed sets it remains to show that it is much faster to compute geodesic closed sets in outerplanar graphs than in arbitrary graphs. Thus, we now give the details of step (ii), i.e., we present a *fast* algorithm that it is in fact *independent* of the input size $|X|$ solving the following problem for *outerplanar* graphs:

Problem 6.1.1. *Given a graph G and $X \subseteq V(G)$, compute $\gamma(X)$.*

The algorithm solving Problem 6.1.1 for outerplanar graphs is given in Algorithm 6 (see, also, Figures 6.2–6.4 for a running example). We assume that G is *connected* by

Algorithm 6: Outerplanar Graphs: Closure

```
   Input: outerplanar graph $G$ and $X \subseteq V(G)$
   Output: $\gamma(X)$
1  construct the BB-tree $\widetilde{G}$ for $G$;
2  $X_0 \leftarrow X$ ;
3  $Y \leftarrow$ set of block nodes of $\widetilde{G}$;
4  $C_1 = \{v_B \in Y : V(B) \cap X_0 \neq \emptyset\}$;
5  $C_2 \leftarrow V(\widetilde{G}) \cap X_0$ ;
6  $C \leftarrow \tau(\widetilde{G}, C_1 \cup C_2), ;$
7  $X_1 \leftarrow X_0 \cup (C \cap V(G)), i \leftarrow 1$;
8  foreach $v_B \in Y \cap C$ do
9      if $|V(B) \cap X_i| > 1$ then
10         $X_{i+1} \leftarrow X_i \cup \beta(B, V(B) \cap X_i)$;
11         $i \leftarrow i + 1$;
12 return $X_i$;
```

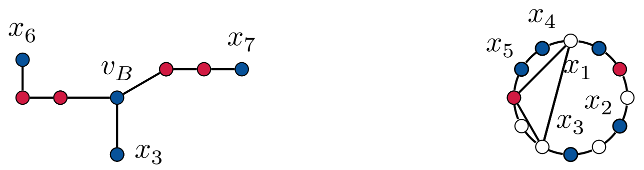


Figure 6.3: *Left:* Output of Algorithm 7 applied to the BB-tree $\tilde{G}$ from Figure 6.2 (nodes in $X_1 \setminus X_0$ are marked in red). *Right:* Biconnected outerplanar graph B corresponding to v_B, nodes in $X_1 \cap V(B)$ which are not in X_0 are marked in red.

noting that all results can easily be generalized to disconnected outerplanar graphs as well. Algorithm 6 first calculates the BB-tree $\tilde{G}$ for the input outerplanar graph G and then stores X and the set of block nodes of $\tilde{G}$ in the variables X_0 and Y, respectively (Lines 1–3). In Line 4, it computes the set C_1 of block nodes representing such blocks of G that have at least one vertex from X_0. In a similar way, C_2 contains the set of nodes of $\tilde{G}$ that belong to X_0 (cf. Line 5) (see Figure 6.2 for an example).

The closure of $C_1 \cup C_2$ in $\tilde{G}$ is calculated in C (Line 6) and the union of X_0 and the set of vertices in C that belong to $V(G)$ is stored in X_1 (Line 7) (see Figure 6.3 for an example). Note that at this point of the algorithm, we have $v \in X_1 \subseteq \gamma_G(X)$ for all $v \in \gamma_G(X)$ that do not belong to a biconnected component of G. Furthermore, for all $v \in \gamma_G(X) \setminus X_1$, v is on a shortest path in one of the blocks and with both endpoints in X. Accordingly, in loop 8–11, the algorithm takes all block nodes v_B of $\tilde{G}$ that belong to

Algorithm 7: Function τ

```
   Input: tree $T$ and $X \subseteq V(T)$
   Output: $\gamma_T(X)$
1  while $\exists v \in V(T) \setminus X$ with $d(v) \leq 1$ do
2      remove $v$ from $T$;
3  return $V(T)$;
```

Algorithm 8: Function GeneratorSet

```
   Input: biconnected outerplanar graph $B$, $X \subseteq V(B)$
   Output: $G_X \subseteq X$ such that $\gamma_B(G_X) = \gamma_B(X)$
1  $G_X \leftarrow \emptyset$                // $G_X \subseteq X$: generator set for $\gamma_B(X)$ ;
2  forall interior faces $F$ of $B$ do
3      $X' \leftarrow V(F) \cap X$;
4      if $|X'| > 0$ then
5          select an arbitrary vertex $w$ from $X'$;
6          add $u = \arg\max_{x \in X'} d(x, w)$ to $G_X$ ;
7          add $v = \arg\max_{x \in (X' \setminus \gamma_F(\{u,w\})) \cup \{w\}} d(x, w)$ to $G_X$ ;
8          if $w \notin \gamma_F(\{u, v\})$ then
9              add $w$ to $G_X$ ;
10 return $G_X$;
```

the closed set C, computes the closure of the set of vertices of the corresponding block B over B that are known to be closed (i.e., belong to X_i), updates the set of already known closed vertices in X_{i+1}, and increments the loop variable i. In the end, it returns the set X_i.

It remains to discuss functions τ and β (cf. Lines 6 and 10). Regarding τ (see Algorithm 7), it computes the closure of a set of nodes of a tree. The algorithm iteratively removes all leaves of T that are not in X and returns the set of all nodes of T at the end that have not been deleted (see Figure 6.2 (right) and Figure 6.3 (left)). Hence, the proof of the following lemma is straightforward:

Lemma 6.1.4. *For any tree T with n nodes and for any $X \subseteq V(T)$, Algorithm 7 returns $\gamma_T(X)$ in $O(n)$ time.*

Regarding β (see Algorithm 9 and Figure 6.4), which computes the closure over biconnected outerplanar graphs, we first show that for any biconnected outerplanar graph B with $f = \Phi(B)$ and for any $X \subseteq V(B)$, there is a set $G_X \subseteq X$ of cardinality *linear* in f such that $\gamma_B(G_X) = \gamma_B(X)$. Furthermore, G_X can be constructed in *linear* time as follows (see, also, Algorithm 8): Initialize G_X with $\emptyset$ (cf. Line 1) and process all interior faces F of B one by one in an arbitrary order as follows: If F has no vertex from X then disregard F; o/w choose an arbitrary vertex w from $X' = V(F) \cap X$. For that w,

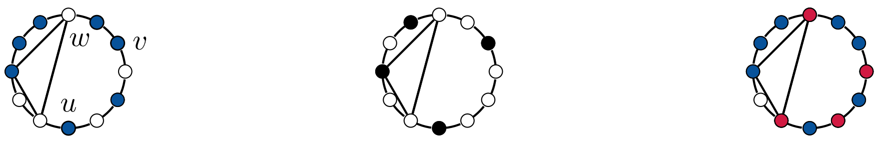


Figure 6.4: *Left:* Biconnected outerplanar graph B with $X \subseteq V(B)$ in blue (cf. Figure 6.3 (right)), *Middle:* generator set $G_X \subseteq X$ in black, *Right:* $\gamma(X) = \gamma(G_X)$, newly added nodes in red.

calculate the furthest vertex $u \in X'$ and the furthest vertex $v \in (X' \setminus \gamma_F(\{u, w\})) \cup \{w\}$, and add u and v to G_X (cf. Lines 6 and 7 of Algorithm 8). Note that $\gamma_F(\{u, w\})) = V(F)$ if $d(u, w) = \ell/2$, where ℓ is the (cycle) length of F; o/w it is the set of vertices of the (unique) shortest path between u and w. If w does not lie on a shortest path between u and v (cf. Line 8), then add w to G_X as well. Note that u and v can be equal to w. Hence, we add at least one and at most three vertices of X' to G_X for F.

To illustrate the above steps on our running example, consider the biconnected outerplanar graph B and the set $X \subseteq V(B)$ marked with the color blue in Figure 6.4. A generator set G_X computed for the input set marked with blue in Figure 6.4 (left) is given in Figure 6.4 (middle). It contains four vertices marked with black. In case of the largest face of G, suppose we first select $w \in X$. For w, we first add u and then v to G_X by Algorithm 8 (see Figure 6.4 (left) for u, v, and w); w is not added because it is on a shortest path between u and v. The closure $\gamma(X) = \gamma(G_X)$ is given in Figure 6.4 (right).

We have the following result about Algorithm 8:

Lemma 6.1.5. *Let B be a biconnected outerplanar graph with $f = \Phi(B)$. Then for all $X \subseteq V(B)$, Algorithm 8 computes a set $G_X \subseteq X$ in $O(n)$ time such that $\gamma_B(G_X) = \gamma_B(X)$ and $|G_X| = O(f)$.*

Proof. Since $G_X \subseteq X$, $\gamma_B(G_X) \subseteq \gamma_B(X)$ follows from the monotonicity of γ_B. We show $\gamma_B(X) \subseteq \gamma_B(G_X)$ by induction on f. The base case $f = 1$ is trivial if B has at most two vertices from X. Otherwise, let u, v, w be the vertices considered by Algorithm 8 for $F = B$. Since $|V(B) \cap X| \geq 2$, we have $u \neq w$. If $v = w$, then $G_X = \{u, v\}$ and $X \subseteq \gamma_B(G_X)$, from which the monotonicity and idempotency of γ_B imply $\gamma_B(X) \subseteq \gamma_B(G_X)$. If $v \neq w$, then u, v, and w are pairwise different. Furthermore, by definition of this case, w does not lie on the (unique) shortest path between u and v. However, then $X \subseteq V(B) = \bigcup_{x,y \in G_X} \gamma_B(\{x, y\})) = \gamma_B(G_X)$, where the last equality holds by Theorem 2.2.1. from which $\gamma_B(X) \subseteq \gamma_B(G_X)$ follows, again by monotonicity and idempotency. For the induction step, let B be a biconnected outerplanar graph with interior faces $F_1, \ldots, F_{f+1}$ for some $f \geq 1$. We can assume w.l.o.g. that $F = F_{f+1}$ is adjacent to exactly one interior face. Then $F_1, \ldots, F_f$ form a biconnected outerplanar graph B'. Let $X_1 = X \cap V(B')$ (resp. $X_2 = X \cap V(F)$) and G_{X_1} (resp. G_{X_2}) be the generator set constructed for B'

```
Algorithm 9: Function $\beta$
  Input: biconnected outerplanar graph $B$, $X \subseteq V(B)$
  Output: $\gamma_B(X)$
1 $G_X \leftarrow$ GeneratorSet$(B, X)$;
2 return $\gamma_B(G_X)$;
```

(resp. F) by Algorithm 8. Note that $G_X = G_{X_1} \cup G_{X_2}$. Theorem 2.2.1 implies

$$\gamma_B(X) = \gamma_B(X_1) \cup \gamma_B(X_2) \cup \bigcup_{u \in X_1, v \in X_2} \gamma_B(\{u, v\}) \ . \tag{6.8}$$

We have

$$\gamma_B(X_1) \subseteq \gamma_{B'}(G_{X_1}) = \gamma_B(G_{X_1}) \subseteq \gamma_B(G_X) \tag{6.9}$$
$$\gamma_B(X_2) \subseteq \gamma_F(G_{X_2}) \ = \gamma_B(G_{X_2}) \subseteq \gamma_B(G_X) \tag{6.10}$$

by $\gamma_B(X_1) = \gamma_{B'}(X_1)$, $\gamma_{B'}(G_{X_1}) = \gamma_B(G_{X_1})$ and $\gamma_B(X_2) = \gamma_F(X_2)$, $\gamma_F(G_{X_2}) = \gamma_B(G_{X_2})$, and by the induction hypothesis to B' and F. Below we show that for all $u \in X_1, v \in X_2$,

$$\gamma_B(\{u, v\}) \subseteq \bigcup_{x, y \in G_X} \gamma_B(\{x, y\}) = \gamma_B(G_X) \ , \tag{6.11}$$

from which $\gamma_B(X) \subseteq \gamma_B(G_X)$ follows by (6.8)–(6.10). To prove (6.11), let $u \in X_1, v \in X_2$, and let F_u be the interior face of B containing u. By construction, there are $u_1, u_2 \in G_{X_1}$ such that $u \in \gamma_{B'}(\{u_1, u_2\})$ and $\gamma_{B'}(\{u_1, u_2\}) \cap G_{X_1} = \{u_1, u_2\}$. Similarly, there are $v_1, v_2 \in G_{X_2}$ with $v \in \gamma_F(\{v_1, v_2\})$ and $\gamma_F(\{v_1, v_2\}) \cap G_{X_2} = \{v_1, v_2\}$. It holds that for all shortest paths $P_{u,v}$ between u and v, there is a shortest path $P_{u',u} \oplus P_{u,v} \oplus P_{v,v'}$ that contains $P_{u,v}$, where $\oplus$ denotes the path concatenation operation and $P_{u',u}$ (resp. $P_{v,v'}$) is a shortest path from some $u' \in \{u, u_1, u_2\}$ to u (resp. from v to some $v' \in \{v, v_1, v_2\}$). One can easily check that $V(P_{u',u}) \subseteq \gamma_B(\{u', u\})$ and $V(P_{v,v'}) \subseteq \gamma_B(\{v, v'\})$ are both subsets of $\gamma_B(G_X)$, from which (6.11) holds by $V(P_{u',v'}) \subseteq \gamma_B(G_X)$, completing the proof of $\gamma_B(X) \subseteq \gamma_B(G_X)$.

The linear time complexity of Algorithm 8 follows from the facts that each iteration of the loop can be carried out in $O(|V(F)|)$ time and the sum of the sizes of the faces F is $O(n)$. □

We are ready to present Algorithm 9 computing the closure of a set of vertices over a biconnected outerplanar graph (see Line 10 in Algorithm 6). The input of Algorithm 9 consists of a biconnected outerplanar graph B and a set $X \subseteq V(B)$. Using Algorithm 8, it first computes a generator set G_X for B and X and, utilizing the results in Corollary 4.2.1, computes $\gamma_B(X) = \gamma_B(G_X)$ in time $O(|V(B)| \cdot |G_X|)$. Hence, we can deduce the following result.

Lemma 6.1.6. *Let B, f, and X be as in Lemma 6.1.5. Then Algorithm 9 computes $\gamma_B(X)$ correctly and in $O(|V(B)| f)$ time.*

In order to state Theorem 6.1.2, the main result of this section, we need some further notation. For any $v \in V(\tilde{G})$, $\Gamma(v)$ denotes $\{v\}$ if $v \in V(G)$; o/w $\Gamma(v) = V(B)$, where B is the block of G represented by v. Proposition 6.1.1 below is used in the proof of Theorem 6.1.2. Its proof follows from the definitions.

Proposition 6.1.1. *Let G be an outerplanar graph and $x \in V(G)$.*

(i) *Let $\tilde{G}$ be the BB-tree of G and $u, v \in V(\tilde{G})$. If $x \in V(\tilde{G})$, then x is on the shortest path in $\tilde{G}$ between u and v if and only if it is on a shortest path in G between u' and v', for all $u' \in \Gamma(u)$ and $v' \in \Gamma(v)$.*

(ii) *Let B be some block of G and $u, v \in V(B)$. Then x is on a shortest path in B connecting u and v if and only if it is on a shortest path in G between u and v.*

Using Lemmas 6.1.4–6.1.6, we can now show the following result:

Theorem 6.1.2. *Algorithm 6 solves Problem 6.1.1 for outerplanar graphs correctly and in $O(nf)$ time, where $f = \Phi(G)$.*

Proof. Regarding the correctness, for the X_is in Algorithm 6 we have $X = X_0 \subseteq X_1 \subseteq \ldots \subseteq X_N \subseteq V(G)$ (cf. Lines 2, 7 and 10), where N is the value of i at termination. We show that $X_N = \gamma(X)$. This is straightforward for $|X| \leq 1$, so assume $|X| > 1$. We prove the soundness (i.e., $X_N \subseteq \gamma(X)$) by showing with induction on i that $X_i \subseteq \gamma(X)$ for all i. The proof of the case $i = 0$ is automatic by the extensivity of γ. For $i = 1$, the same argument holds if $x \in X_0$, so consider the case that $x \in X_1 \setminus X_0$. Then, by Lemma 6.1.4 concerning the correctness of τ computing the closure over trees (cf. Line 6), x belongs to the closure of $C_1 \cup C_2$ in $\tilde{G}$. That is, there are $u, v \in C_1 \cup C_2 \subseteq V(\tilde{G})$ such that x lies on a shortest path connecting u and v in $\tilde{G}$, from which we have $x \in \gamma(X)$ by (i) of Proposition 6.1.1. For the induction step, suppose $X_k \subseteq \gamma(X)$ holds for $k \geq 1$ and let $x \in X_{k+1}$. If $x \in X_k$, then $x \in \gamma(X)$ by the induction hypothesis. Otherwise, by the definition of k, x has been added to X_{k+1} in Line 10. However, then, $x \in \gamma(X)$ is immediate from (ii) of Proposition 6.1.1 by Lemma 6.1.6 concerning the correctness of β computing the closure over biconnected outerplanar graphs, and by the induction hypothesis, completing the proof of soundness.

For the completeness (i.e., $\gamma(X) \subseteq X_N$), let $x \in \gamma(X)$. Clearly, $x \in X_N$ if $x \in X$. Otherwise, by Theorem 2.2.1, there are $u, v \in X$ with $u \neq v$ such that $x \in \gamma(\{u, v\})$. If x does not belong to a block in G, then $x \in V(\tilde{G})$ and $\Gamma(u) \neq \Gamma(v)$. Let $u', v' \in V(\tilde{G})$ such that $u \in \Gamma(u')$ and $v \in \Gamma(v')$. We must have that x, u', v' are pairwise different. But then, by (i) of Proposition 6.1.1, x is on a shortest path in $\tilde{G}$ that connects u' and v' and it has been added to X_1, as $u', v' \in C_1 \cup C_2$ by definition. Consider the case that $x \in V(B)$ for some block B of G. Let P be a shortest path in G with endpoints u and v that contains x. If $u, v \in V(B)$, then the node $v_B \in V(\tilde{G})$ representing B has been added to Y in Line 3 and processed in loop 8–11. In particular, x is added to X_{i+1} for some $i \geq 1$ because $u, v \in V(B) \cap X_i$ (cf. Line 10). If at least one of u, v is not a vertex of B, then let $u_\perp, v_\perp \in V(B)$ be the vertices on P with the smallest distance to u and v, respectively. The definitions imply that $u_\perp, v_\perp \in V(\tilde{G})$. Furthermore, $u_\perp, v_\perp \in X_1$ by (i) of Proposition 6.1.1 and Lemma 6.1.4. We are done if $x = u_\perp$ or $x = v_\perp$. Otherwise, x is on a shortest path between $u_\perp$ and $v_\perp$ in B and hence, as $u_\perp, v_\perp \in X_1 \subseteq X_i$, it is added to X_{i+1} for some $i \geq 1$ in Line 10 for v_B, as β is correct by Lemma 6.1.6. Hence, $\gamma(X) \subseteq X_N$.

Regarding the complexity, $\tilde{G}$ in Line 1 can be computed in $O(n)$ time Horváth et al. (2010) and, by Lemma 6.1.4, the closure operator τ over $\tilde{G}$ (cf. Line 6) can be calculated also in $O(n)$ time. Suppose G contains k blocks, say $B_1, \ldots, B_k$. Since by Lemma 6.1.6, the closure operator β over B_i (cf. Line 10) can be computed in $O(n_i f_i)$ time for all i, where $n_i = |V(B_i)|$ and $f_i = \Phi(B_i)$, loop 8–11 can be carried out in $\sum_i O(n_i f_i) = O(nf)$ time, as $\sum_i O(n_i) = O(n)$ and $f_i = O(f)$. Thus, the total time of Algorithm 6 is $O(nf)$, as claimed. □

6.1.3 Experimental Results

Finally, after the detailed elaboration of steps (i) and (ii) in the previous sections, we experimentally evaluate the steps and the heuristic by approximating geodesic cores of large real-world graphs. Thus, the experiments are separated into three parts. First, we evaluate Algorithm 4 for its runtime and quality of the returned outerplanar spanning subgraphs. We also compare its runtime against other standard algorithms generating spanning *subtrees*. Second, we compare the runtime of our outerplanar closure algorithm (Algorithm 6) to that of the naïve algorithm for outerplanar graphs (see Section 2.2.2). Finally, using *large real-world* networks (Leskovec and Krevl, 2014), we empirically evaluate the approximation performance of our heuristic on the *core-periphery* decomposition problem (Marc and Šubelj, 2018). For the implementation[1] we used the C++-library Snap 6.0 (Leskovec and Sosič, 2016). All experiments were conducted on a machine with AMD Ryzen 9 3900X and 64GB RAM.

Sampling spanning structures

In these experiments, we empirically evaluate the runtime and performance of Algorithm 4 on graphs from the Erdős-Rényi II dataset. For each of the 100 random graphs for a particular value of (n, p), we first generate a spanning structure with different algorithms and then compare the average generation runtime for the 100 graphs (see Figure 6.5). The following algorithms have been considered: (O1) is Algorithm 4, (SBFS) implemented in Snap 6.0 generates a spanning BFS-tree, and (BFS) and (DFS) are our own implementations generating spanning BFS resp. DFS trees. We also consider (O2), which first calls (O1) and then calculates the BB-tree and the biconnected outerplanar components for the output outerplanar graphs of (O1). The reason for considering (O2) is that our closure algorithm described in Section 6.1.2 also requires these additional pieces of information.

In Figure 6.5 (left) we compare the runtime of the algorithms with respect to the edge number of the graphs in the Erdős-Rényi II dataset. The results show that (O1) and (O2)

[1]The code is available at `https://github.com/fseiffarth/GCoreApproximation`.

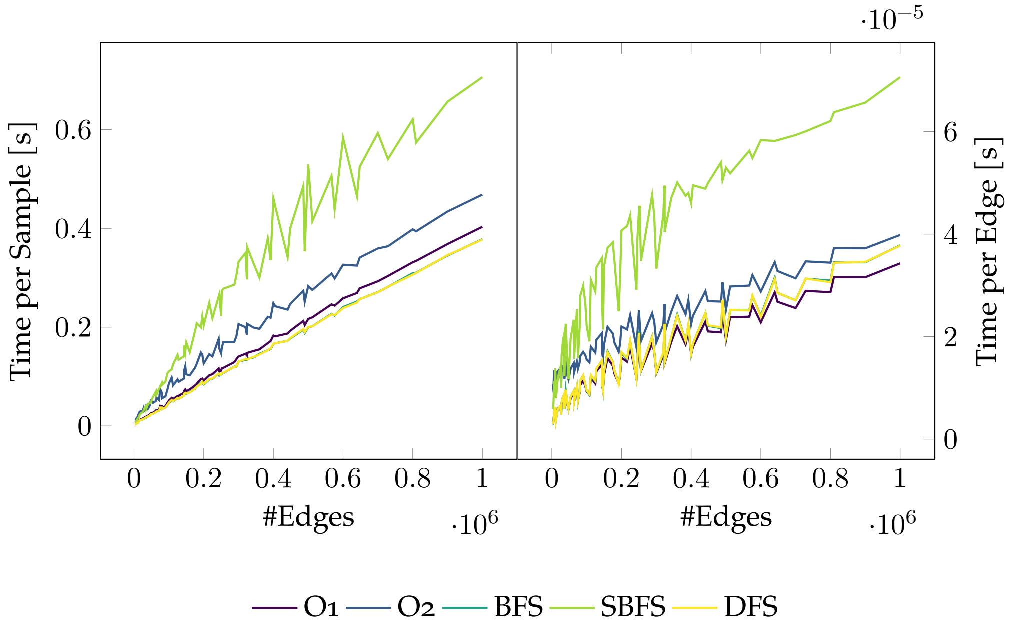


Figure 6.5: Average time per sample (in sec.) of generating spanning structures with different algorithms on the Erdös-Renyi II dataset. (left) Average time per sample and output edge number in seconds. (right) The considered algorithms: (O1) Algorithm 4, (O2) Algorithm 4 + generating the auxiliary information for Algorithm 6, (SBFS) BFS algorithm in SNAP 6.0, (BFS) resp. (DFS) own BFS resp. DFS implementations

are linear in the number of edges and almost as fast as (BFS) and (DFS), by noting that (SBFS) is even slower than our algorithm generating outerplanar spanning subgraphs. Not surprisingly, (O2) is a bit slower than (O1) but remains linear, as the additional information provided by (O2) can be calculated in linear time. Figure 6.7 shows that this is not a drawback in practice because the auxiliary structure generated by (O2) allows for a much faster closure computation. For graphs with around 10^4 nodes and 10^6 edges, (BFS) needs $0.38s$ per spanning *tree*, while (O1) resp. (O2) $0.4s$ resp. $0.47s$ per spanning *outerplanar* graph. Normalizing the runtime by the number of output edges, (O1) is *faster* than (BFS) and (DFS) (see Figure 6.5 (right)).

More detailed information about the generated outerplanar spanning subgraphs for the Erdős-Rényi II dataset is provided in Table 6.1 respectively Table 6.2 for random graphs with $n = 10^4$ respectively $p = 0.2$. The average number of edges in the output outerplanar spanning subgraphs seems to grow nearly *linearly* with the edge probability. The average number of biconnected components, as well as the average maximal size of the components, decreases, i.e., for small edge probabilities, it seems that there are a few big components. At the same time, there are more medium size components and no big components for larger density.

Recall that the time complexity of our closure algorithm presented in Section 6.1.2

Edge Prob.	#Edges	Sample Size	Avg. #Output Edges	Avg. #Components	Avg. Biggest Component	Avg. Face Number
0.006	299,970	100	10,922.11 ($\pm$ 20.71)	638.68 ($\pm$ 63.84)	828.33 ($\pm$ 265.95)	79.44 ($\pm$ 25.54)
0.008	399,960	100	11,077.61 ($\pm$ 19.76)	595.81 ($\pm$ 60.51)	677.62 ($\pm$ 224.51)	76.11 ($\pm$ 25.38)
0.010	499,950	100	11,214.71 ($\pm$ 21.43)	585.54 ($\pm$ 69.05)	591.26 ($\pm$ 176.29)	74.50 ($\pm$ 23.16)
0.012	599,940	100	11,342.36 ($\pm$ 23.01)	580.80 ($\pm$ 59.00)	508.81 ($\pm$ 119.71)	70.52 ($\pm$ 16.36)
0.014	699,930	100	11,454.92 ($\pm$ 25.05)	584.37 ($\pm$ 56.59)	478.12 ($\pm$ 131.24)	71.92 ($\pm$ 20.65)
0.016	799,920	100	11,561.69 ($\pm$ 25.60)	591.34 ($\pm$ 60.36)	454.45 ($\pm$ 123.27)	71.77 ($\pm$ 19.26)
0.018	899,910	100	11,659.78 ($\pm$ 21.80)	590.60 ($\pm$ 52.77)	389.05 ($\pm$ 74.31)	65.98 ($\pm$ 13.10)
0.020	999,900	100	11,755.85 ($\pm$ 27.71)	590.04 ($\pm$ 48.96)	370.94 ($\pm$ 80.23)	65.95 ($\pm$ 14.14)

Table 6.1: Output of Algorithm 4 on Erdős-Rényi II random graphs with fixed size of $n = 10^4$.

Size	#Edges	Sample Size	Avg. #Output Edges	Avg. #Component	Avg. Biggest Component	Avg. Face Number
1,000	9,990	100	1,154.42 ($\pm$ 8.33)	128.34 ($\pm$ 17.81)	209.91 ($\pm$ 71.51)	35.98 ($\pm$ 12.14)
2,000	39,980	100	2,332.69 ($\pm$ 11.84)	181.91 ($\pm$ 25.29)	263.11 ($\pm$ 81.09)	46.16 ($\pm$ 14.79)
3,000	89,970	100	3,506.71 ($\pm$ 13.18)	234.28 ($\pm$ 30.47)	300.53 ($\pm$ 99.33)	52.21 ($\pm$ 16.42)
4,000	159,960	100	4,683.94 ($\pm$ 15.39)	280.25 ($\pm$ 31.77)	312.23 ($\pm$ 77.90)	55.34 ($\pm$ 13.39)
5,000	249,950	100	5,859.74 ($\pm$ 18.30)	335.01 ($\pm$ 36.19)	320.67 ($\pm$ 86.55)	57.17 ($\pm$ 15.86)
6,000	359,940	100	7,039.75 ($\pm$ 20.77)	387.48 ($\pm$ 43.64)	323.84 ($\pm$ 89.57)	57.86 ($\pm$ 16.21)
7,000	489,930	100	8,218.86 ($\pm$ 20.38)	438.71 ($\pm$ 42.61)	346.45 ($\pm$ 86.02)	61.86 ($\pm$ 14.56)
8,000	639,920	100	9,397.93 ($\pm$ 23.03)	482.03 ($\pm$ 47.00)	358.90 ($\pm$ 74.91)	63.98 ($\pm$ 13.65)
9,000	809,910	100	10,579.86 ($\pm$ 24.38)	546.62 ($\pm$ 52.48)	374.12 ($\pm$ 92.46)	66.53 ($\pm$ 16.23)
10,000	999,900	100	11,755.85 ($\pm$ 27.71)	590.04 ($\pm$ 48.96)	370.94 ($\pm$ 80.23)	65.95 ($\pm$ 14.14)

Table 6.2: Output of Algorithm 4 on Erdős-Rényi II dataset with fixed edge probability $p = 0.02$.

is *linear* in the face number of the input outerplanar graph. Table 6.2 and Figure 6.6 indicate that in practice, the face number seems to be sublinear in the graph size for fixed density. In our experiments, it was always less than 80. Figure 6.6 shows the number of nodes (left), respectively the number of input edges (right) against the average face number (colors represent edge probabilities).

Moreover, we experimentally show that in all cases, the output of Algorithm 4 is not far from a *maximal* outerplanar subgraph. It was tested by greedily adding edges to the output as long as there exists an edge whose addition does not violate outerplanarity. Since the test needs $O(m)$ time, for these experiments, we used the Erdős-Rényi I dataset containing small graphs. In Table 6.3 resp. Table 6.4 we present the results for fixed size $n = 500$ resp. fixed edge probability $p = 0.14$. We expect a similar behavior on larger graphs because the main factor of the input graphs is their density and not their size. For the non-maximal outerplanar graphs, it holds that at most 2 edges on average are missing per graph for maximality (see Table 6.3 for $n = 500$ and Table 6.4 for $p = 0.14$). On the relative scale, the outerplanar subgraphs returned by our algorithm contain at least 99.71% of the edges of a *maximal* spanning outerplanar subgraph on average (see the last column of Tables 6.3 and 6.4). The standard deviation of these values is always smaller than 0.5 and less than 0.2 in most cases, implying that there are only a few outliers.

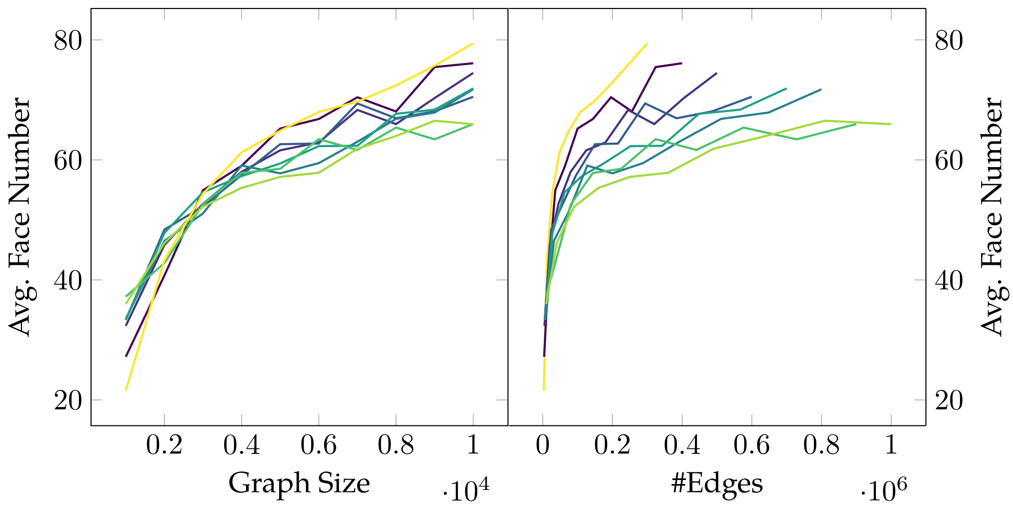


Figure 6.6: Face numbers of outerplanar subgraphs generated by Algorithm 4 for the Erdős-Rényi II dataset. (left) Average face number against the graph size. (right) Average face number against input edge number (colors depict edge probabilities).

Calculating closures in outerplanar graphs

In this section, we empirically evaluate Algorithm 6 on synthetically generated data. More precisely, we sample an outerplanar spanning subgraph G for each graph in the Erdős-Rényi II dataset. To compare our algorithm with the standard one based on SSSP, for each outerplanar graph G, we construct a graph G' with the same number of nodes and edges, but with the difference that G' is not necessarily outerplanar. That is, for each G, we first generate a random spanning tree T of G and then construct a possibly non-outerplanar graph G' by adding $m - n + 1$ random edges to T. Thus, G and G' have the same number of vertices and edges. Figure 6.7 (left) shows the average runtime needed to calculate the closures on G and G' for a random subset of 1% of the vertices. (C1) is the naïve closure algorithm for the outerplanar graphs G using the result of Corollary 4.2.1 (i.e., it calculates the shortest paths between all pairs of input vertices). (C2) is our Algorithm 6 and (CGraph) is the naïve closure algorithm for the arbitrary graphs G'. Recall that the complexity of (CGraph) is $O(nm)$, where $m = O(n)$ by construction, it is $O(n|X|)$ for (C1), where $|X| = n/100$, and $O(nf)$ for our algorithm (C2), which is independent of $|X|$. The results are in accordance with these complexities. In particular, the closure computation on the arbitrary graphs G' is slower by a factor up to 300 than on the outerplanar graphs G with (C1) and (C2) (see left of Figure 6.7). The right part of Figure 6.7 is scaled down for (C1) and (C2). It clearly shows that (C2) (i.e., Algorithm 6) is much faster in practice than the naïve algorithm (C1). In particular, (C2) seems to be the only one of the three algorithms that scales *linearly* with the number of edges. This indicates that the face number f in the time complexity $O(nf)$ is *negligible* in practice.

Edge Prob.	#Edges	Samples (Maximal)	Avg. #Output Edges	Avg. Maximal Output Edges	Avg. Relative Maximality (%)
0.05	6,237	100 (32)	627.14 ($\pm$ 5.85)	628.30 ($\pm$ 5.90)	99.82 ($\pm$ 0.17)
0.06	7,485	100 (24)	641.76 ($\pm$ 6.69)	643.14 ($\pm$ 6.96)	99.79 ($\pm$ 0.18)
0.07	8,732	100 (25)	655.67 ($\pm$ 7.58)	657.13 ($\pm$ 7.74)	99.78 ($\pm$ 0.19)
0.08	9,980	100 (21)	670.08 ($\pm$ 7.61)	671.68 ($\pm$ 7.84)	99.76 ($\pm$ 0.19)
0.09	11,227	100 (17)	680.30 ($\pm$ 6.29)	681.85 ($\pm$ 6.43)	99.77 ($\pm$ 0.16)
0.10	12,475	100 (17)	691.20 ($\pm$ 8.33)	693.08 ($\pm$ 8.26)	99.73 ($\pm$ 0.22)
0.11	13,722	100 (24)	703.77 ($\pm$ 8.47)	705.44 ($\pm$ 8.63)	99.76 ($\pm$ 0.20)
0.12	14,970	100 (12)	715.28 ($\pm$ 8.56)	717.24 ($\pm$ 8.68)	99.73 ($\pm$ 0.21)
0.13	16,217	100 (20)	724.40 ($\pm$ 8.47)	726.05 ($\pm$ 8.75)	99.77 ($\pm$ 0.18)
0.14	17,465	100 (14)	731.81 ($\pm$ 8.56)	733.71 ($\pm$ 8.72)	99.74 ($\pm$ 0.18)

Table 6.3: Quality of the output of Algorithm 4 on Erdős-Rényi I for fixed $n = 500$.

Size	#Edges	Samples (Maximal)	Avg. Output Edges	Avg. Maximal Output Edges	Avg. Relative Maximality (%)
100	693	100 (69)	139.45 ($\pm$ 3.34)	139.86 ($\pm$ 3.43)	99.71 ($\pm$ 0.48)
200	2,786	100 (41)	287.66 ($\pm$ 5.29)	288.48 ($\pm$ 5.37)	99.72 ($\pm$ 0.31)
300	6,279	100 (37)	437.00 ($\pm$ 5.58)	438.03 ($\pm$ 5.66)	99.77 ($\pm$ 0.25)
400	11,172	100 (26)	584.22 ($\pm$ 7.75)	585.56 ($\pm$ 7.77)	99.77 ($\pm$ 0.19)
500	17,465	100 (14)	731.81 ($\pm$ 8.56)	733.71 ($\pm$ 8.72)	99.74 ($\pm$ 0.18)

Table 6.4: Quality of the output of Algorithm 4 on Erdős-Rényi I for fixed $p = 0.14$.

This observation is supported by Table 6.1. It reports the average *face number* of the generated spanning outerplanar subgraphs for the graphs with $n = 10^4$ vertices in the Erdős-Rényi II dataset. Somewhat surprisingly, the average face number does *not* increase with the density. Figure 6.6 shows the average face number as a function of the number of vertices (left) and the number of edges of the input graphs (right), where the colors represent different edge probabilities. The results indicate that in practice, the face number seems to be *sublinear* in the graph size for fixed density (in our experiments, it was always less than 80), justifying the *better* runtime of our closure computation algorithm (see Figure 6.7 (right)).

Approximation of convex cores in large real-world graphs

Finally, applying the heuristic described above, we present experiments concerning the approximation of *geodesic cores* in *large real-world* networks. We shortly recall the definition of geodesic cores provided in Section 2.4. The geodesic core $\mathcal{C}$ of a graph G is defined by $\bigcap_{j=1}^{i} C_j$, where i is the smallest integer satisfying $\bigcap_{j=1}^{i} C_j = \bigcap_{j=1}^{i+1} C_j$ and $C_j = \gamma(X_j)$ is the closure of $X_j \subseteq V(G)$ containing $l > 0$ vertices selected independently and uniformly at random. Note that this definition is not deterministic, but our and also experiments by Marc and Šubelj (2018) show that if a core exists, then it is *stable*, i.e., the choice of X_j and especially l does not affect the core if l is sufficiently large. In particular, as a compromise between runtime and stability with respect to random effects, we choose $l = 10$. For each of the networks in Table 6.5, the fixed point was reached after $i = 3$ iterations.

We used 15 networks from (Leskovec and Krevl, 2014) in our experiments. The size (n)

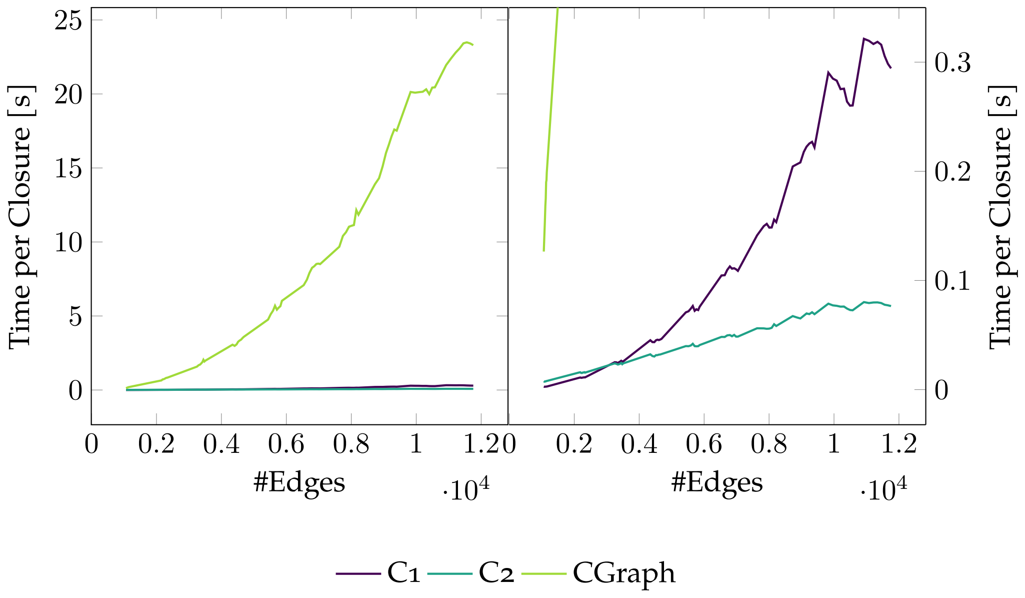


Figure 6.7: Closure runtimes for outerplanar graphs (C1), (C2) resp. non-outerplanar graphs (CGraph) with the same number of nodes and edges. The generator set is a random subset of 1% of the vertices. (C1) is the naive algorithm, (C2) is Algorithm 6.

and order (m) of some of them are more than $1{,}000$ times larger than those in (Marc and Šubelj, 2018). Table 6.5 contains the size of the exact cores and the runtime of computing them. While the *exact* core of the 3 largest networks could not be computed within 50 days using the standard algorithm to compute geodesic closed sets sketched in Section 2.2.2, our algorithm produced the *approximate* cores in 5 h for these large networks; in less than 40 min for all other graphs.

For the approximation, for each large network, we generated $s = 100$ spanning outerplanar subgraphs and calculated the closure of l randomly chosen vertices on each of these outerplanar graphs with Algorithm 6. Given the 100 closed sets in the outerplanar subgraphs obtained this way, a vertex $v \in G$ was regarded as closed if and only if it was contained in at least t out of the $s = 100$ closed sets. The approximate core $\widetilde{\mathcal{C}}$ was then calculated in the same iterative way as the exact one (see Section 2.4), but using the approximate closed sets instead of the exact ones. We compared exact and approximate cores with each other using Jaccard similarity. The first value in the last column of Table 6.5 denotes the *best* Jaccard similarity achieved via a grid search over $l \in \{5, \dots, 2000\}$ and $t \in \{1, \dots, 10\}$. We stress that using higher values of l has *no* impact on the time complexity of our algorithm, as it depends on n and the face number only (cf. Section 6.1.2). The second value (in brackets) denotes the Jaccard similarity for the approximate core obtained for $l = 5$ and $t = 1$ averaged over 10 runs.

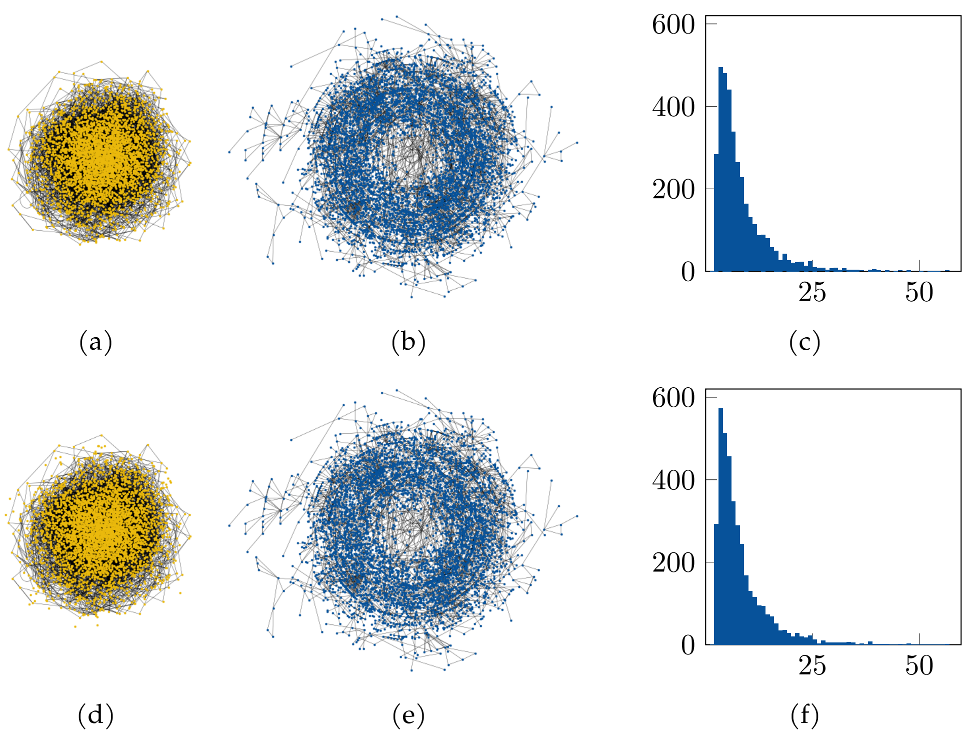


Figure 6.8: CA-HepTh network, its exact (a) core, (b) periphery, (c) degree distribution of the core and its approximated (d) core, (e) periphery, (f) degree distribution of the approximate core.

For 12 out of the 15 graphs, we obtained an average Jaccard similarity of around 0.8 or more; for 9 even at least 0.9. As an example, in Figure 6.8, we show the exact core and periphery of the CA-HepTh network (see (a) and (b)) and their approximations (see (d) and (e)) for $l = 5$ and $t = 1$ (see, also, Table 6.5). We also plot the degree distribution of the exact core (see (c)) and that of the approximate core (see (f)) obtained for these values. One can see that the two distributions are relatively similar to each other by noting that the Jaccard similarity obtained for $l = 5$ and $t = 1$ was 0.93 (see Table 6.5). A similar behavior could be observed for the other networks as well.

Graph	Size	#Edges	Density	Size	#Edges	Time [s]	Approx.	Time [s]	Jaccard similarity
	n	m		Core	Core	Exact	Core	Approx.	best ($l = 5, t = 1$)
com-Orkut	3,072,441	117,185,083	2.5e-05	n.a.	n.a.	n.a.	2,915,420	1.8e+04	n.a.
soc-LiveJournal1	4,843,953	43,362,750	3.7e-06	n.a.	n.a.	n.a.	3,018,149	8.7e+03	n.a.
soc-pokec-relationships	1,632,803	22,301,964	1.7e-05	n.a.	n.a.	n.a.	1,390,297	6.5e+03	n.a.
com-youtube.ungraph	1,134,890	2,987,624	4.6e-06	390,825	2,169,158	8.9e+05	338,654	2.2e+03	0.82 (0.71)
com-dblp.ungraph	317,080	1,049,866	2.1e-05	90,077	438,265	7.0e+04	92,833	5.3e+02	**0.92** (0.87)
com-amazon.ungraph	334,863	925,872	1.7e-05	216,109	643,075	2.2e+05	231,618	5.2e+02	0.88 (0.87)
Slashdot0902	82,168	582,533	1.7e-04	48,718	514,338	1.4e+04	45,558	1.6e+02	**0.92** (0.71)
Cit-HepPh	34,401	420,828	7.1e-04	32,111	417,050	6.1e+03	32,309	9.6e+01	**0.99** (**0.97**)
Cit-HepTh	27,400	352,059	9.4e-04	24,832	347,918	3.5e+03	25,049	7.7e+01	**0.98** (**0.98**)
CA-AstroPh	17,903	197,031	1.2e-03	9,487	142,943	6.4e+02	9,522	3.0e+01	**0.95** (**0.93**)
CA-CondMat	21,363	91,342	4.0e-04	8,603	49,682	4.0e+02	8,761	3.5e+01	**0.94** (**0.90**)
CA-HepPh	11,204	117,649	1.9e-03	4,825	63,548	1.8e+02	4,804	1.8e+01	**0.93** (**0.91**)
Wiki-Vote	7,066	100,736	4.0e-03	4,579	98,026	1.3e+02	4,452	1.5e+01	**0.97** (0.75)
CA-HepTh	8,638	24,827	6.7e-04	3,605	14,161	4.6e+01	3,669	1.2e+01	**0.96** (**0.93**)
CA-GrQc	4,158	13,428	1.6e-03	1,336	5,036	7.0e+00	1,380	6.0e+00	**0.92** (0.88)

Table 6.5: Large real-world networks from Leskovec and Krevl (2014) with number of vertices (n), number of edges (m), density, number of vertices and edges in the core, time to calculate the exact core in seconds (or n.a. if it was not possible within 50 days), size of the approximated core (the result of grid search), time to calculate the approximated core, and the Jaccard similarities of the exact and approximated cores obtained by grid search over l (number of random nodes in the generator set) and t (frequency threshold for considering a node to be an element of the approximate core), and for $l = 5, t = 1$ in brackets (average over 10 runs) (values of at least 0.9 in bold). The networks are sorted by nm.

Graph	Approx. Core (mean)	Approx. Core (normalized std.)	Jaccard Similarity
Wiki-Vote_component	3,453.8	3.65e-02	0.75
com-youtube.ungraph_component	287,435.0	3.17e-02	0.71
Slashdot0902_component	34,561.9	1.53e-02	0.71
CA-HepPh_component	5,071.0	7.52e-03	0.91
com-amazon.ungraph_component	230,612.7	6.79e-03	0.87
com-dblp.ungraph_component	100,934.7	5.33e-03	0.87
CA-GrQc_component	1,487.4	5.21e-03	0.88
CA-CondMat_component	9,401.1	4.91e-03	0.90
Cit-HepPh_component	31,328.6	3.70e-03	0.97
CA-AstroPh_component	9,704.1	2.56e-03	0.93
CA-HepTh_component	3,825.3	1.33e-03	0.93
Cit-HepTh_component	24,485.1	1.32e-03	0.98

Table 6.6: Comparison of the normalized standard deviation over 10 different approximate geodesic cores and the Jaccard similarity achieved. The column *Approx. Core (Mean)* denotes the mean size over ten approximations of the geodesic core. The column *Approx. Core (normalized std.) denotes the corresponding standard deviation divided by the mean value. The Jaccard Similarity denotes the mean similarity between the approximated and exact geodesic cores.* The table is sorted according to the normalized standard deviation.

Regarding the quality of the approximation of the geodesic core, it is of course difficult to evaluate if if the exact core cannot be calculated. In fact, we could not evaluate the quality of our approximation for the three largest graphs depicted in Table 6.5. One possibility to evaluate the quality of our approximation without exactly computing the geodesic core is to analyze the variance of the size of the approximated geodesic cores considered over different approximation runs. In particular, for each of the graphs depicted in Table 6.6, we run our heuristic 10 times and compute the standard deviation of the sizes of the approximate cores. We then normalize the standard deviation by dividing it by the mean size of the approximated geodesic cores. For the 12 graphs from Table 6.5 for which we know the approximation quality, Table 6.6 shows the mean size of the approximated cores and the normalized standard deviation over 10 runs of our approximation algorithm (for parameters $l = 5$ and $t = 1$). The table, sorted by the normalized standard deviation, shows that the Jaccard similarity between approximate and exact cores increases with decreasing deviation. Indeed, for a normalized deviation of more than 0.0153, we reach a Jaccard similarity of less than 0.75. For a normalized deviation of less than 0.005, we achieve a Jaccard similarity above 0.90. We note that this apriori analysis affects the runtime only a little, as the time consuming part is the sampling which only has to be done once. After sampling, each run of our approximation heuristic can be done in time linear in the size of the graph.

6.2 A Simple Heuristic for the Graph Tukey Depth

As a second practical application, we present an analysis of the concept of graph Tukey depth to mining and learning with graphs. *Centrality measures* are crucial in data analysis, as they typically capture the elements' "importance" quantitatively. Of course, the meaning of *importance* depends on the choice of the particular centrality measure. Different types of centrality measures have been introduced for networks (see, e.g., Newman (2018)), including *degree centrality*, *eigenvector centrality*, *Katz centrality*, *closeness centrality*, *betweenness centrality*, *page rank*, and *hubs and authorities*. In Figure 6.9, we present a graphical illustration of some of these centrality measures and the graph Tukey depth for some small graphs for a visual comparison. Moreover, since graph Tukey depth is based on geodesic closures, we show that besides measuring the centrality of vertices, it allows for interesting associations to the previously discussed theory and concepts. Thus, before turning to the algorithmic aspects of the *graph Tukey depths* problem (cf. Section 2.5 for the definition), in the next section we first present some of its *potential* applications to mining and learning with graphs.

6.2.1 Graph Tukey Depth: Potential Applications to Mining and Learning with Graphs

In this section, we will raise some interesting connections of graph *Tukey depth* to *vertex separations* and *geodesic core-periphery decompositions*. Thus, we state three essential properties of Tukey depth. In particular, Proposition 6.2.1 clarifies Tukey depth's role in geodesic closed sets.

Proposition 6.2.1. *Let $G = (V, E)$ be a graph, $v \in V(G)$ with $\mathrm{td}(v) = |V(G)| - c$, and $C \subseteq V(G)$ a geodesically closed vertex set with $|C| > c$. Then $v \in C$.*

Proposition 6.2.2 (Cerdeira and Silva (2021))**.** *Let $G = (V, E)$ be a graph, $X \subseteq V(G)$, and $C = \gamma(X)$ be the geodesic closure of X. Then the graph Tukey depth is a quasi-concave function, i.e., for all $c \in C$ we have $td(c) \geq \min\{td(x) : x \in X\}$.*

Proposition 6.2.3 (Cerdeira and Silva (2021))**.** *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$, and $X = \{v \in V(G) : \mathrm{td}(v) \geq k\}$. Then X is geodesically closed.*

To underline the importance of these three statements, we give two examples that show how they (can) influence machine learning and data mining methods based on geodesic closures.

Example 1: Vertex Classification and Active Learning In Section 5.4 and (de Araújo et al., 2019; Thiessen and Gärtner, 2021; Thiessen and Gärtner, 2022), disjoint half-spaces and closed sets are used for binary classification in closure systems, for vertex classification, and active and online learning in graphs using geodesic convexity. Given the Tukey depth $\mathrm{td}(v)$ of a vertex v, Proposition 6.2.1 immediately implies that a separating half-space or closed set not containing v cannot have a cardinality greater than $|V(G)| -$

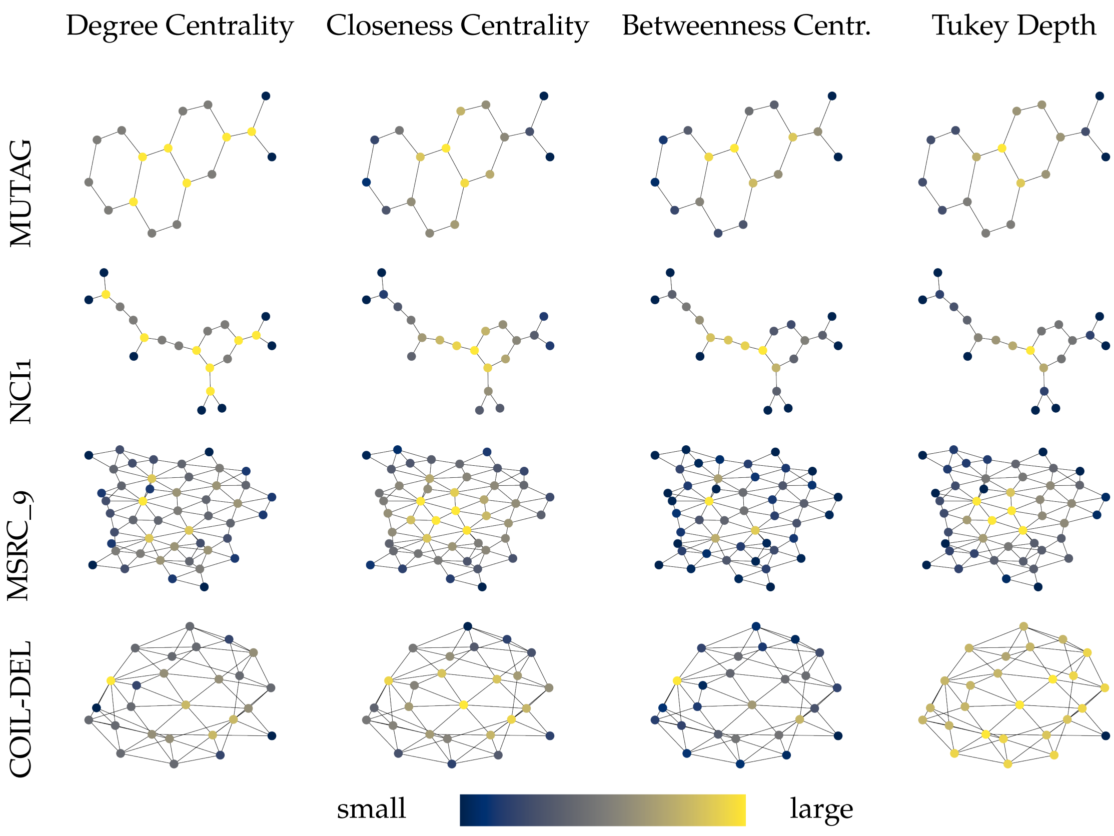


Figure 6.9: The *Degree Centrality*, *Closeness Centrality*, *Betweenness Centrality* and *Tukey Depth* of vertices in graphs selected from different graph datasets (Morris et al., 2020). The centrality (resp. depth) values are *normalized* (i.e., mapped to the interval $[0, 1]$) by their maximum values in the graph. In particular, vertices of the smallest (resp. highest) centrality values are denoted by blue (resp. yellow).

td(v). Thus, for vertices of *high* Tukey depth, there is *no* large geodesic closed set *not* containing them. Hence, Proposition 6.2.1 implies a nice theoretical connection between graph Tukey depth and the maximum size of separating half-spaces and closed sets. Using (approximate) graph Tukey depths, the predictive performance of the above methods can possibly be affected and improved.

Example 2: Geodesic core-periphery decomposition In Section 6.1, we considered the geodesic core-periphery decomposition of graphs (Marc and Šubelj, 2018; Šubelj et al., 2019). In particular, they found out that many social networks consist of a dense geodesic core "surrounded" by a sparse periphery (see Figure 2.7 for an example). While some graphs, especially tree-like graphs, seem to have no core, others, such as graphs sampled from random graph models (e.g., Erdős-Rényi, Barabási-Albert and Watts-Strogatz) seem to have no periphery. Moreover, the closure of a small number of randomly chosen graph vertices (≈ 10) always contains the geodesic core (if it exists). Furthermore, if the vertices are sampled from the geodesic core only, then the closure of the vertices is the geodesic core itself. If we compute the closure of, say, 10 randomly chosen vertices from the entire network (Figure 2.7(a)), then the closure always contains the core (orange vertices in Figure 2.7(b)). If all random vertices belong to the core (orange vertices in Figure 2.7(a)), then their closure is the core itself. The above statements explain this behavior. Using that the core is always contained in the closure of a small number of randomly chosen vertices, from Proposition 6.2.2 it follows that the vertices in the core are those with the highest Tukey depths. Moreover, the quasi-concave property of graph Tukey depth implies that if the core is generated by a few vertices from the core, then the core vertices must have a very close Tukey depth. Finally, using Proposition 6.2.3, we have that the set of vertices in a graph with a Tukey depth above some threshold is always geodesically closed; geodesic cores arise as a particular case of this property.

The above example and the three properties motivate the following *deterministic* definition of geodesic cores:

Definition 6.2.1. *The k-geodesic core of a graph G is defined by*

$$C_k := \{v \in V(G) : \text{td}(v) \geq k\} \ .$$

To empirically confirm our claim that the core contains the vertices with the highest Tukey depths, we considered the three graphs in Figure 6.10. We computed the exact Tukey depths (top row) and their geodesic cores (middle row). The depths from small to large are visualized by the corresponding color bar. The geodesic cores are marked in yellow, and the periphery in blue. Moreover, we visualized all the vertex Tukey depths as a barplot (sorted by the depth) together with the information on which of the vertices belong to the core (yellow) and which to the periphery (blue) (bottom row).

For the Karate Club network (left), also considered in the work of Cerdeira and Silva (2021), the geodesic core exactly matches the set of vertices of Tukey depth ≥ 19 (see the barplot in the bottom). Hence, in this case the geodesic core defined in Section 2.4 is in fact the k-geodesic core defined above for $k = 19$. Furthermore, there is not much

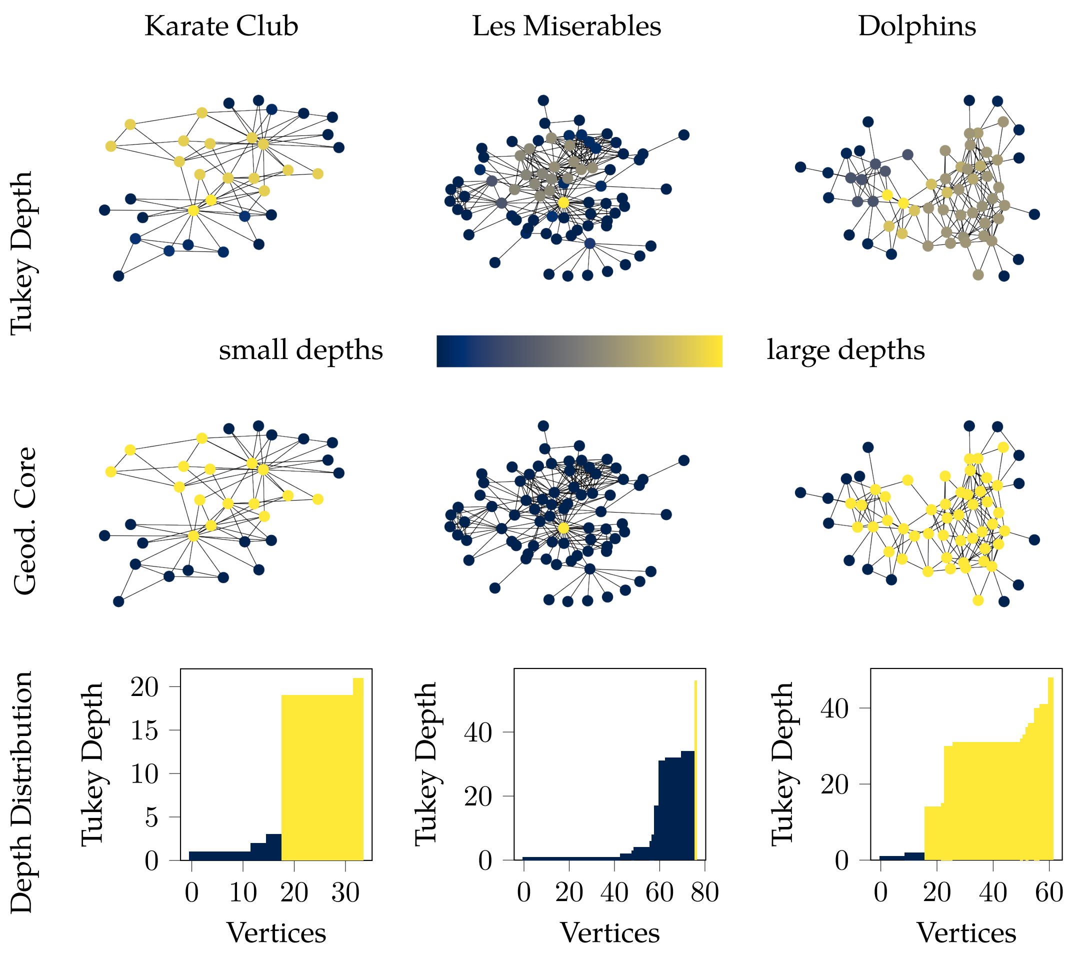


Figure 6.10: Tukey depth (top) vs. geodesic core-periphery decomposition (middle) and distribution of the Tukey depths (bottom) for the Karate Club (Zachary, 1977), Les Miserables character (Knuth, 1993), and Dolphins social networks (Lusseau, 2003). For the different Tukey depths we use sequential colors. Core and periphery vertices are denoted by yellow and blue, respectively. The distribution of Tukey depths shows all the depths sorted by their value. Vertices in the core (respectively in the periphery) are marked by a yellow bar (respectively by a blue bar).

fluctuation in the depths of the core vertices. In fact, all vertices of Tukey depth of at most 3 belong to the periphery, and all vertices of Tukey depth 19 or 21 to the core, by noting that there are *no* vertices of Tukey depth between 4 and 18. In case of the Les Miserables character network (middle), there is only a single vertex with a very high Tukey depth of 57, surrounded by vertices of depth less than 35. In this case, the core algorithm returns only the vertex with the highest Tukey depth. For the Dolphin community graph (right), the geodesic core consists of all vertices with Tukey depth greater than 2, while all vertices in the periphery have a Tukey depth of at most 2. The three examples suggest that the probabilistic definition of geodesic cores by Marc and Šubelj (2018) can be alternatively described by the deterministic k-geodesic core for some number k. Moreover, we claim that the graph Tukey depth can be used to refine core-periphery decompositions.

6.2.2 Approximating the Tukey Depth

Motivated by the negative complexity result concerning the calculation of Tukey depth, in Section 6.2.3 below, we propose a *heuristic* based on Algorithm 1 that solves the *MCSS* problem. It approximates the vertices' Tukey depths with one-sided error. We show experimentally on different types of *small* graphs that the results obtained by our heuristic are *fairly close* to the exact ones. Furthermore, even on small graphs, our algorithm is up to 200 times faster than the exact one (Section 6.2.4). It is important to emphasize that we had to resort to such graphs, as it was not possible to calculate the exact Tukey depths for larger graphs in a practically feasible time.

6.2.3 The Heuristic

Recall that the *exact* Tukey depth of a vertex v is defined by $\mathrm{td}(v) := |V(G)| - |C|$, where $|C|$ is the *maximum* cardinality of a closed set C *not* containing v. It can be computed *exactly* using an *integer linear program* (see Cerdeira and Silva, 2021, for the details). The computationally *hard* part of the problem is to find a closed set of *maximum* size. Our heuristic addresses this problem by considering an inclusion *maximal* closed set only instead of a maximum cardinality closed set. This relaxation, which distorts of course the exact value of Tukey depth, allows us to apply Algorithm 1 for solving the *MCSS* problem. In what follows, for any $v \in V(G)$, $\widetilde{\mathrm{td}}(v)$ denotes the approximation of $\mathrm{td}(v)$ obtained with our heuristic.

Given a graph G, the rough idea to approximate the Tukey depth of a vertex $v \in V(G)$ is to find an inclusion maximal geodesically closed set $C \subseteq V(G)$ with $v \notin C$. Such a set C can be found by applying Algorithm 1 with input sets $\{v\}$ and $\{v'\}$. Then the output of the algorithm is a solution of the *MCSS* problem, i.e., it consists of two closed vertex sets $H_v, H_{v'} \subseteq V(G)$ with $v \in H_v$ and $v' \in H_{v'}$ that are disjoint and inclusion maximal. That is, there exist no proper closed supersets of $H_v, H_{v'}$ with the same properties. The Tukey depth can then be approximated using the cardinalities of H_v resp. $H_{v'}$. For a fixed vertex v, the result depends on the particular choice of v'. To improve the approximation quality, we therefore call the *MCSS* algorithm for each vertex v several times, with *different* vertices $v' \neq v$.

Algorithm 10: Approximation of Graph Tukey Depth

```
Input   : graph $G$
Output  : approximation $\widetilde{\mathrm{td}}(v)$ of $\mathrm{td}(v)$ for all $v \in V(G)$
1  $\widetilde{\mathrm{td}}(v) \longleftarrow |V(G)|$ for all $v \in V(G)$;
2  forall $v \in V(G)$ do
3      forall $v' \in \Gamma(v)$ do
4          $H_{v'}, H_v = MCSS(\{v'\}, \{v\})$;
5          forall $x \in V(G)$ do
6              if $x \notin H_{v'}$ then
7                  $\widetilde{\mathrm{td}}(x) = \min\{\widetilde{\mathrm{td}}(x), |V(G)| - |H_{v'}|\}$;
8              if $x \notin H_v$ then
9                  $\widetilde{\mathrm{td}}(x) = \min\{\widetilde{\mathrm{td}}(x), |V(G)| - |H_v|\}$;
10 return $\widetilde{\mathrm{td}}(v)$ for all $v \in V(G)$
```

The pseudo-code of the above heuristic is given in Algorithm 10. In Line 1 we initialize the Tukey depth of all vertices in G by setting them to the maximum possible value, i.e., to $|V(G)|$. We repeat the procedure described above for all vertices $v \in V(G)$ and all their neighbors $v' \in \Gamma(v)$ (see the for-loops in Line 2 and 3). In this way we solve the *MCSS* problem for all input sets $\{v\}, \{v'\}$, i.e., separate v from all of its neighbors v' by maximal disjoint closed sets $H_v, H_{v'}$ (see Line 4). Note that the Tukey depth of a vertex x is based on a closed set of *maximum* cardinality not containing x. Thus, if x does not lie in the set H_v (resp. $H_{v'}$), then the cardinality of H_v (resp. $H_{v'}$) is smaller than or equal to a closed set of maximum cardinality not containing x. Hence, we can update the current Tukey depth approximation of *all* vertices $x \in V(G)$ as follows: Take the minimum over the old and the new approximation which is the cardinality of $V(G) \setminus H_v$ if $x \notin H_v$ or that of $V(G) \setminus H_{v'}$ if $x \notin H_{v'}$ (see Line 7 and Line 9).

By construction, Algorithm 10 finds only *maximal* and *not* maximum closed sets, resulting in a one-sided error in the estimate of Tukey depths. The result is formulated in the proposition below.

Proposition 6.2.4. *Algorithm 10 overestimates the Tukey depth, i.e., for the output $\widetilde{\mathrm{td}}(v)$ returned by Algorithm 10 we have $\widetilde{\mathrm{td}}(v) \geq \mathrm{td}(v)$, for all $v \in V(G)$.*

Regarding the runtime of Algorithm 10, note that the inner part of the loop in Lines 3–9 is executed $O(m)$ times because we iterate over all neighbors (i.e., all edges are considered twice). The runtime of the inner loop (Lines 3–9) is dominated by the *MCSS* algorithm called in Line 4, which calls the closure operator at most $O(n)$ times (see Theorem 5.1.1). Since the geodesic closure can be computed in time $\mathcal{O}(mn)$ (Pelayo, 2013), we have the following result for the total runtime of Algorithm 10:

Proposition 6.2.5. *Algorithm 10 returns an upper bound of the Tukey depth for all vertices of G in $\mathcal{O}(m^2n^2)$ time.*

The runtime of the approximation algorithm can be improved by considering for each vertex v a *fixed* number of distinct vertices v', or by considering a fixed subset $W \subseteq V(G)$, instead of the whole vertex set $V(G)$ in the outer loop (see Lines 2–9). It is left to further research to analyze how these changes affect the quality of the approximation performance.

6.2.4 Experimental Evaluation

In this section, we empirically evaluate the approximation quality and runtime of Algorithm 10 on datasets containing *small* graphs[2]. Regarding the approximation quality, we compare the results obtained by our heuristic algorithm to the exact Tukey depths computed with the algorithm proposed by Cerdeira and Silva (2021). For the evaluation, we consider 19 graph datasets by (Morris et al., 2020) of different types (small molecules, small graphs from bioinformatics and computer vision, and small social networks). See columns 2–4 of Table 6.7 for the number of graphs and their average number of vertices and edges. The *average size* of the graphs ranges from 14 (*PTC_MM*) up to 82 (*OHSU*); their *average edge numbers* from 14 to 200. The reason for considering small graphs only is that the exact algorithm (Cerdeira and Silva, 2021) was unable to calculate the Tukey depth for larger graphs within one day (see the last two columns of Table 6.7). For practical reasons, we removed all disconnected graphs from the original datasets by noting that our heuristic also works for disconnected graphs.

The results are presented in Table 6.7. It contains the approximation qualities measured in different ways (columns 5–10) and the runtime of the exact (column 11), and our heuristic algorithm (column 12). The datasets are sorted according to their *absolute approximation error* (column 5 of Table 6.7), i.e., the sum of all differences between the approximation and the exact Tukey depth over all vertices and all graphs in the dataset.

Regarding the *absolute error* (column 5), our approximation results are equal to the exact Tukey depths for 5 out of the 19 datasets, while their computation was faster by a factor of up to 100 (see *PTC_MM*). Our algorithm has the largest absolute error of 4155 on the *COIL-DEL* graphs, by noting that this dataset consists of 3900 graphs. Hence, the average error per graph is only slightly above one. Additionally, we look at the *relative errors* (column 6), i.e., the absolute error divided by the sum of all depths. We use this measure to validate that our algorithm performs very well, by noting that the relative errors are below $4 \cdot 10^{-3}$ for all graph datasets. The *per vertex error* (column 7) is the average error our algorithm makes per vertex, while the *per graph error* (column 8) is the error it has on average per graph. Regarding the *per vertex error*, the worst-case is for the *COIL-DEL* dataset (last row) with an average error of 0.05. For the *per graph error*, the worst result was obtained for the *OHSU* dataset, where we overestimate the sum of all vertex depths by 1.65 per graph on average. The results show that our approximation algorithm performs very well, especially if considering the average results over the datasets.

[2]See https://github.com/fseiffarth/AppOfTukeyDepth for the code.

Finally, we also studied the worst-case approximations for vertices and graphs. In particular, the columns *Max. Vertex Error* respectively *Max. Graph Error* denote the *maximum error* of the algorithm on single vertices respectively on single graphs. The results show a very low error of at most 3 per vertex for 13 out of the 19 datasets. For three graph datasets, the *maximum error per vertex* is at most 7, and we have a maximum error between 11 and 19 in three cases. Regarding the *maximum error per graph*, a similar behavior can be observed by noting that except for *OHSU* and *Peking_1*, the *maximum vertex errors* and *maximum graph errors* are close to each other. This implies that there are only a *few* vertices with a *high* approximation error. It is an interesting problem to find the structural properties of such vertices and graphs responsible for the high approximation errors. The last two columns show the *runtimes* of the two algorithms. Our algorithm (last column) is *faster* than the exact one by at least *one order* of magnitude. In summary, the evaluation of Algorithm 10 clearly demonstrates that our heuristic performs well in approximating the graph Tukey depth. It is faster (sometimes more than 100 times) than the exact algorithm, even on small graph datasets. Regarding larger graphs, this gap in runtime will increase because of the exponential runtime of the exact algorithm. Additionally, the very small relative errors (at most $4 \cdot 10^{-3}$), the average errors (at most 1.65 per graph), and also the worst case errors show that the algorithm can be used effectively for further applications based on the Tukey depth (see Section 6.2.1).

Data	Graph Number	Avg. Vertices	Avg. Edges	Error (absolute)	Error (relative)	Error per Vertex	Error per Graph	Max. Vertex Error	Max. Graph Error	Exact Runtime (s)	Approx. Runtime (s)
BZR	405	35.75	38.36	0	0	0	0	0	0	56.60	1.04
PTC_MM	336	13.97	14.32	0	0	0	0	0	0	21.09	0.21
COX2	467	41.22	43.45	0	0	0	0	0	0	76.10	1.27
Cuneiform	267	21.27	44.80	0	0	0	0	0	0	2.00	0.61
DHFR	756	42.43	44.54	0	0	0	0	0	0	266.33	3.19
PTC_FR	351	14.56	15.00	1	4.50e-05	1.96e-04	2.85e-03	1	1	23.81	0.25
PTC_FM	349	14.11	14.48	1	4.80e-05	2.03e-04	2.86e-03	1	1	20.64	0.22
MUTAG	188	17.93	19.79	1	6.50e-05	2.97e-04	5.32e-03	1	1	3.01	0.09
PTC_MR	344	14.29	14.69	2	9.20e-05	4.07e-04	5.81e-03	1	1	23.29	0.23
KKI	83	26.96	48.42	12	1.01e-03	5.36e-03	1.45e-01	2	4	40.45	2.43
IMDB-BINARY	1000	19.77	96.53	19	4.37e-04	9.61e-04	1.90e-02	1	3	723.07	113.14
NCI1	3530	29.27	31.88	34	5.20e-05	3.29e-04	9.63e-03	6	8	1194.63	7.76
Peking_1	85	39.31	77.35	40	1.30e-03	1.20e-02	4.71e-01	3	10	4761.33	48.08
MSRC_21C	209	40.28	96.60	86	1.28e-03	1.02e-02	4.11e-01	12	12	51.78	3.06
MSRC_9	221	40.58	97.94	89	1.21e-03	9.92e-03	4.03e-01	7	8	49.33	2.92
OHSU	79	82.01	199.66	130	8.54e-04	2.01e-02	1.65e+00	2	13	42887.32	235.40
ENZYMES	569	31.68	61.44	307	1.68e-03	1.70e-02	5.40e-01	7	10	933.79	8.25
MSRC_21	563	77.52	198.32	877	1.39e-03	2.01e-02	1.56e+00	19	25	3679.24	58.98
COIL-DEL	3900	21.54	54.24	4155	4.05e-03	4.95e-02	1.07e+00	11	17	2242.05	43.00

Table 6.7: Graph data of different sizes selected from Morris et al. (2020). Disconnected graphs are removed from the original datasets. The columns regarding the approximation quality denote the following. *Error (absolute)* denotes the overall error on the dataset, *Error (relative)* denotes the relative error regarding the depths, *Error per Vertex* denotes the average error per vertex, *Error per Graph* denotes the average error per graph, *Max. Vertex Error* denotes the maximum error for a vertex and *Max. Graph Error* denotes the maximum error on a graph. The last two columns show the runtimes of the exact and approximation algorithm in seconds.

6.3 Summary and Open Questions

We summarize the main results of Section 6.1 and Section 6.2 and discuss some open questions and further research directions.

Approximating Geodesic Closures and Geodesic Cores The main goal of Section 6.1 is to give a fast approximation of geodesic closed sets in large real-world networks. We have proposed a three-step heuristic that can be summarized as follows. Sample spanning subgraphs of the input graph, compute the closed sets in the samples instead of the input graph, and use these closed sets to approximate the original closure. For the first step, we decided to sample spanning *outerplanar* graphs. Our experimental results clearly demonstrate that the presence of *cyclic* edges in the spanning subgraphs is essential for a close approximation of the geodesic convex hull. One of the natural questions is whether other graph classes *beyond* forests can also be considered for spanning subgraphs. Such a graph class should fulfill at least two properties: (i) A (potentially maximal) spanning subgraph from this class could be generated in time *linear* in the order of the input graph, and (ii) for all graphs in this class, the preclosure of any set of vertices should be its closure at the same time (cf. Theorem 2.2.1). This second condition indicates that the graphs in the class should be $K_{2,3}$-free (with respect to forbidden minor). A somewhat related question is whether the algorithm presented in Section 6.1.1 can be modified in a way that it returns a *maximal* spanning outerplanar graph, preserving at the same time the time complexity of Algorithm 4. A positive answer to this question would provide an *algorithmic* solution to the result of Djidjev (2006).

Since our primary goal is to find the best possible approximation of geodesic closed sets and geodesic cores, it is a question if we even need maximal outerplanar subgraphs. There may exist particular edges that are more important than others and hence should be added to the outerplanar subgraphs. For example, is it possible to utilize the *degree distribution* of the input graph in the selection of the back edges in a way that the output outerplanar graphs give better approximation results? Or, is it possible to generate the spanning subgraphs such that each edge of the input graph is contained in at least one of the subgraphs?

Our empirical results concerning core approximation in large real-world networks have been obtained for relatively small sets of generator elements and for low frequency thresholds. The choice of these two parameters seem crucial for a close approximation (see Table 6.5). The related question is how to select them, especially in case of large networks. *Sampling* seems a natural way, the question is whether it is possible to utilize the structure of the network at hand during sampling. Last but not least, it would be engaging to *systematically* study other types of random as well as large real-world networks for their core-periphery decomposition. One step towards a better understanding of the core-periphery decomposition structure of different types of graphs is given in Section 6.2. Instead of the *probabilistic* definition by Marc and Šubelj (2018) and also in Section 2.4, we could give a *deterministic* definition using the graph Tukey depth.

Graph Tukey Depth in Mining and Learning with Graphs Graph Tukey depth is an exciting and promising new concept for mining and learning with graphs. We have shown that by definition, the graph Tukey depth is closely related to half-space and other types of closed set separations, as it grasps the separability of an element. Our results clearly indicate, that the depth of an element is an essential measure for supervised learning on graphs (Section 5.4.2) as well as for active (Thiessen and Gärtner, 2021) and online learning (Thiessen and Gärtner, 2022) on graphs. We show that it is possible to relate the graph Tukey depth to the geodesic core-periphery decomposition that was discussed in detail in Section 6.1 by mentioning that the vertices in the core are exactly those of high graph Tukey depth. Our result shows that graph Tukey depth or more general, the definition of Tukey depth in arbitrary finite closure systems (Section 2.5) strongly connects the importance of elements with closed set separations. Hence, it seems an essential tool for mining and learning methods.

The study of the relationship between graph Tukey depth and other vertex centrality measures is an interesting question for further research (cf. Figure 6.9). For example, while the centroid(s) in trees (Piotrowski, 1987; Piotrowski and Syslo, 1991) are precisely the vertices with the highest Tukey depth, this is not necessarily the case for graphs beyond trees.

Another issue is a *better understanding* of the semantics behind the graph Tukey depth, especially its quasi-concaveness property (Proposition 6.2.2). For example, what are the properties of the vertices with the highest depth (cf. the definition of Tukey-median in $\mathbb{R}^d$)? We have empirically demonstrated that graph Tukey depth can closely be approximated in small graphs. It is an open question whether this result holds also for (very) large graphs. To answer this question, the scalability of our approximation algorithm should be improved on the one hand. On the other hand, one needs (possibly tight) theoretical upper bounds on graph Tukey depths. Another interesting question is to identify graph classes for which our heuristic always results in the *exact* graph Tukey depth. While this is the case for trees, it is unclear whether it also holds, e.g., for outerplanar graphs. We believe this question can be answered affirmatively using techniques similar to those presented in Section 6.1.2. We have experimentally shown that there is a connection between geodesic core-periphery decompositions and the deterministic definition of k-geodesic cores. On the one hand, we can explain the nature of the geodesic core-periphery decomposition by analyzing graph Tukey depth. On the other hand, the connection enables us to use our core approximation algorithm introduced in Section 6.1 to approximate the set of vertices with the highest Tukey depth.

7 Concluding Remarks

In this chapter, we discuss the main results of the thesis and formulate some open questions and problems for future works.

7.1 Discussion

One of our main research questions was to study the following problem: To what extent is it possible to adapt linear separability from $\mathbb{R}^d$ to abstract finite closure systems. Our primary goal was to derive a *general* separation method that applies to *concept learning* in finite closure systems and assumes as little domain-specific knowledge as possible. In fact, we have aimed at a *domain-independent* separation algorithm that only relies on a *domain dependent* closure operator. In other words, our goal was to design a learning (or separation) algorithm for finite closure systems that has access to the closure system via the corresponding closure operator only. To achieve this goal, we have first discussed the problem of half-space separation in finite closure systems and have shown that it does not possess the basic properties of the well-known half-space separation problem in $\mathbb{R}^d$. In particular, in case of finite closure systems, two disjoint closed sets are not necessarily separable by half-spaces, i.e., the result of Kakutani (1937) for $\mathbb{R}^d$ does not hold. We have referred to this problem by formulating the half-space separation (HSS) problem and have shown that it is even hard to *decide* if there exists a separating half-space for two disjoint closed sets. Since we were interested in practically feasible algorithms, we have relaxed the original problem of half-space separations. In particular, we have considered two alternative solutions. First, we have looked at *specific* closure systems called Kakutani closure systems. Second, we have *simplified* the problem to the maximal closed set separation (MCSS) problem.

Regarding Kakutani closure systems, we have considered the specific case of geodesic closure systems over graphs. We have shown that $K_{2,3}$ free graphs and hence, outerplanar graphs are always Kakutani closure systems. For solving the MCSS problem, we have proposed a simple greedy algorithm (Algorithm 1) that provides maximal closed set separations for *arbitrary* disjoint closed sets. Somewhat surprisingly, this simple algorithm is optimal with respect to the number of closure operator calls. In fact, our result demonstrates that without incorporating domain-specific knowledge, it is *not* possible to perform better than this simple greedy algorithm. Thus, in a follow-up step, we have analyzed how to improve the basic greedy algorithm by incorporating *additional* knowledge. We have distinguished between *domain-specific* (e.g., that arises from the particular structure of the domain) and *domain-independent* knowledge (e.g., distance-based knowledge). The results of Section 5.2.2 show that Algorithm 1 can be improved in terms of the number of closure operator calls by utilizing the particular structural properties of *lattices*. Hence, it would be interesting to study specializations of Algorithm 1 to other domains, particularly to special graphs and relational structures.

Incorporating domain-independent knowledge and adapting the developments of linear separations in machine learning in $\mathbb{R}^d$, we have generalized some main ideas of support vector machines to finite closure systems. In particular, while the perceptron algorithm (Rosenblatt, 1958) considers *arbitrary* hyperplane separations, support vector machines (Boser et al., 1992) find the *maximum margin* hyperplane by taking into account the distances between elements in the underlying metric space. We have transferred this idea to separations in finite closure systems by "enriching" the underlying ground set with an abstract "distance" function between elements and sets by using *monotone linkage functions* (Mullat, 1976). In this way, we have extended the simple greedy algorithm that provides arbitrary maximal closed set separations to a version that finds maximum margin maximal closed set separations in finite closure systems. We have shown empirically that this knowledge integration enhances the predictive performance compared to the simple baseline greedy algorithm.

As a summary of our theoretical results, we regard linear separation problems in finite closure systems as maximal closed set separation problems for the following reasons: In general, there exists no half-space separation of disjoint closed sets. In case of Kakutani closure systems, any solution to the maximal closed set separation problem provides a half-space separation at the same time. Moreover, we have shown that domain-specific knowledge can be used to improve the greedy algorithm with respect to the number of closure operator calls and the performance of supervised learning tasks in different finite closure systems.

Regarding the practical aspects of our results, for geodesic closure systems over graphs we have given a straightforward heuristic that closely approximates geodesic closed sets in large graphs. The main steps of this heuristic may be of some independent theoretical interest. In particular, we have given a *fast* algorithm for finding outerplanar spanning subgraphs that are *almost* maximal and developed a fast algorithm that computes the closure operator on outerplanar graphs. Besides these algorithmic results, we have applied the heuristic to approximate geodesic cores. This approximation allows for an analysis of geodesic core-periphery decompositions of *large* real-world graphs. Furthermore, we have raised some fundamental relationships between closed set separations, geodesic core-periphery decompositions, and graph Tukey depths. Finally, we have shown that our greedy Algorithm 1 can be used not only as a basis for domain-specific algorithms, but also to approximate the Tukey depth of vertices in relatively small graph databases.

7.2 Outlook

Our results presented in the thesis raise several open questions that are of theoretical and practical interest. Throughout this work, we have assumed that the closure operator is given by an oracle that returns the closure of a set *extensionally*. In case of domain-specific closure systems (e.g., lattices), closed sets (e.g., ideals and filters) can, however, be represented *intensionally* (e.g., by their suprema and infima). As another example, for closure systems over trees we have that any half-space has a succinct intensional representation by a single node together with the edge connecting it to the complementary half-space. These and other examples motivate the study of structural properties of closure systems allowing for some compact *intensional* representation of abstract half-spaces and closed sets. In particular, we would aim for such representations of half-spaces and closed sets that allow us to decide if an element is contained in the set or not without enumerating all elements of the set (cf. the case of hyperplane separations in $\mathbb{R}^d$). For example, in case of lattices we only have to compare an element with the corresponding supremum (respectively infimum) of the maximal closed set representation.

We have seen that maximum margin separations are *not* unique, in contrast to maximum margin hyperplanes in $\mathbb{R}^d$. Thus, it is an interesting question to ask for additional properties restricting the possible number of solutions for maximum margin separations. In this thesis, we have adapted several notions from classical machine learning in $\mathbb{R}^d$. An advantage of separations in $\mathbb{R}^d$ is that they are representable by the support vectors. It is an open question whether the somewhat related concept of *support elements* in case of maximum margin separations in finite closure systems fulfills similar properties. Regarding maximum margin separations, what we are particularly interested in is a *formal* description of the gain in the information we can achieve compared to *arbitrary* maximal closed set separations. One way could be to measure the performance improvements in supervised learning tasks and the efficiency of the algorithms with respect to the number of closure operator calls. One direction for future work is to look at theoretical guarantees on these problems. The choice of the monotone linkage function can play an important role when considering maximum margin separations. For example, the monotone linkage function measuring the proximity between feature vectors defined in Section 2.3 has several promising applications. It can be applied to the problem of analyzing maximum margin separations in closed itemset mining by measuring the linkage between features of items and the joint features of a closed itemset.

Throughout this thesis, we have considered *binary* separation problems only. Clearly, they can naturally be extended to *multi-class* separation or *clustering* problems. That is, the binary case can be extended to the problem of finding a k-partitioning of the ground set or k (inclusion) maximal closed sets that are pairwise disjoint, for some $k \geq 2$ integer. While the generalization of our results and Algorithms 1 and 3 concerning maximal closed set separation is straightforward, it is less obvious for the k-partitioning problem, which is also referred to as the k-Kakutani problem (cf. Section 4.2.1). We note that for the particular case of graphs, these problems have already been studied (Artigas et al., 2010, 2011, 2007; Buzatu and Cataranciuc, 2015, 2018).

In this thesis we have assumed that the target concepts are representable by disjoint closed sets. In case of real-world data with noise or outliers, however, we need a less strict problem definition. For such cases, an explicit tolerance of a certain amount of error in the separation process can be beneficial. That is, instead of requiring a disjoint separation of closed sets, one should also allow overlapping closed sets with some bounded overlap size, e.g., by restricting the number of elements present in more than one of the closed sets returned by the algorithm. This kind of problem can be seen as an adaptation of *soft-margin* support vector machines (Cortes and Vapnik, 1995) to finite closure systems. As another alternative to handle noisy data, it would be interesting to look at (closed) fuzzy sets (Hüllermeier, 2011; Lowen, 1980; Nguyen, 1978). Indeed, fuzzy sets are less restrictive than closed sets with respect to containment.

There are several open questions regarding the practical aspects of approximations of geodesic closed sets in large real-world graphs and those of the graph Tukey depth. Some of them have been mentioned in Section 6.3; some will be discussed below. In particular, looking at our heuristic approximating geodesic cores, it is an open question whether we can give some theoretical guarantees for the approximation quality in advance. For example, it is an interesting question whether there are particular graph properties that lead to a better respectively worse approximation quality. In Section 6.2.1 we have shown that geodesic-core periphery decompositions are closely related to the graph Tukey depth. It seems beneficial to study the graph Tukey depth in order to understand this kind of decompositions. Thus, it would be important to study the promising connection between graph Tukey depth and the geodesic-core periphery decomposition. For example, Marc and Šubelj (2018) visualize the growth of certain induced convex subgraphs in different social networks and random graphs. We claim that this growth coincides with the growth of the sorted vertex Tukey depths of a graph.

Looking at graph Tukey depth, we note that each graph convexity defines another centrality measure (based on Tukey depth) for a graph. For example, for the graph convexity relying on shortest paths of length smaller than 5, the corresponding graph Tukey depth is only influenced by the local graph structure, instead of the global one. We note that in this case, each vertex with distance ≥ 5 can be added to a maximum closed set *not* containing v. In particular, the Tukey depth of a vertex v relying on shortest paths of length less than 5 only depends on the (local) ball around v with radius 4. Hence, it would be interesting to analyze the graph Tukey depths of one graph for different convexities (e.g., based on shortest paths of different lengths). Regarding further applications of Tukey depth, note that we have given a general definition that holds for *arbitrary* finite closure systems. It is an interesting research question to consider the depth for closure systems other than geodesic ones. For example, the Tukey depth could be studied in the context of *closed itemset mining* and *formal concept analysis*. The related question could be formulated as follows: What is the Tukey depth of an item or a formal concept and what is the semantics behind this depth? For these and other applications, we need a better understanding of the semantics behind the elements' importance, provided by the centrality measure of Tukey depth in finite closure systems. The original Tukey depth in $\mathbb{R}^d$ provides a centrality measure that is robust against (small) perturbations of the elements and outliers (Dai et al., 2022). It is an interesting question if this robustness property can be generalized to Tukey depths defined over finite closure systems (e.g., geodesic closure systems over graphs).

Bibliography

Adaricheva, K., Nation, J. B., and Rand, R. (2013). Ordered direct implicational basis of a finite closure system. *Discrete Applied Mathematics*, 161(6):707–723.

Allgeier, B. (2009). *Structure and properties of maximal outerplanar graphs*. PhD thesis, University of Louisville.

Anthony, M. and Ratsaby, J. (2016). Multi-category classifiers and sample width. *Journal of Computer and System Sciences*, 82(8):1223–1231.

Anthony, M. and Ratsaby, J. (2018). Large-width bounds for learning half-spaces on distance spaces. *Discretete Applied Mathematics*, 243:73–89.

Anthony, M. and Ratsaby, J. (2020). Large-width machine learning algorithm. *Progress in Artificial Intelligence*, 9(3):275–285.

Araujo, J., Campos, V., Giroire, F., Nisse, N., Sampaio, L., and Soares, R. (2013). On the hull number of some graph classes. *Theoretical Computer Science*, 475:1–12.

Artigas, D., Dantas, S., Dourado, M. C., and Szwarcfiter, J. L. (2010). Convex covers of graphs. *Matemática Contemporânea, Sociedade Brasileira de Matemática*, 39:31–38.

Artigas, D., Dantas, S., Dourado, M. C., and Szwarcfiter, J. L. (2011). Partitioning a graph into convex sets. *Discrete Mathematics*, 311(17):1968–1977.

Artigas, D., Dourado, M. C., and Szwarcfiter, J. L. (2007). Convex partitions of graphs. *Electronic Notes in Discrete Mathematics*, 29:147–151.

Bair, J. (1975). Separation of two convex sets in convexity spaces and in straight line spaces. *Journal of Mathematical Analysis and Applications*, 49(3):696–704.

Banach, S. (1929). Sur les fonctionnelles linéaires. *Studia Mathematica*, 1(1):211–216.

Barwise, J. (1977). An introduction to first-order logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier.

Becker, C. and Gather, U. (1999). The masking breakdown point of multivariate outlier identification rules. *Journal of the American Statistical Association*, 94(447):947–955.

Bennett, K. P. and Campbell, C. (2000). Support vector machines: hype or hallelujah? *ACM SIGKDD explorations newsletter*, 2(2):1–13.

Birkhoff, G. (1940). *Lattice theory*, volume 25. American Mathematical Soc.

Boley, M., Horváth, T., Poigné, A., and Wrobel, S. (2010). Listing closed sets of strongly accessible set systems with applications to data mining. *Theoretical Computer Science*, 411(3):691–700.

Borgatti, S. and Everett, M. (1999). Models of core/periphery structures. *Social Networks*, 21:375–395.

Boros, E., Gurvich, V., Khachiyan, L., and Makino, K. (2003). On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39(3):211–221.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Bremner, D., Chen, D., Iacono, J., Langerman, S., and Morin, P. (2008). Output-sensitive algorithms for tukey depth and related problems. *Statistics and Computing*, 18(3):259–266.

Bressan, M., Cesa-Bianchi, N., Lattanzi, S., and Paudice, A. (2021). Exact recovery of clusters in finite metric spaces using oracle queries. In *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 775–803. PMLR.

Bryant, V. W. and Webster, R. J. (1972). Convexity spaces. I. The basic properties. *Journal of Mathematical Analysis and Applications*, 37(1):206–213.

Bryant, V. W. and Webster, R. J. (1973). Convexity spaces. II. Separation. *Journal of Mathematical Analysis and Applications*, 43(2):321–327.

Bryant, V. W. and Webster, R. J. (1977). Convexity spaces. III. Dimension. *Journal of Mathematical Analysis and Applications*, 57(2):382–392.

Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C. (2016). Recent advances in graph partitioning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9220 LNCS of *LNCS*, pages 117–158. Springer.

Buzatu, R. and Cataranciuc, S. (2015). Convex graph covers. *The Computer Science Journal of Moldova*, 23(3):251–269.

Buzatu, R. and Cataranciuc, S. (2018). On nontrivial covers and partitions of graphs by convex sets. *The Computer Science Journal of Moldova*, 26(1):3–14.

Calder, J. (1971). Some elementary properties of interval convexities. *Journal of the London Mathematical Society*, 2(3):422–428.

Cerdeira, J. O. and Silva, P. C. (2021). A centrality notion for graphs based on tukey depth. *Applied Mathematics and Computation*, 409:126409.

Chartrand, G. and Harary, F. (1967). Planar Permutation Graphs. *Annales de l'institut Henri Poincaré (B) Probabilités et Statistiques*, 3(4):433–438.

Chepoi, V. (1994). Separation of two convex sets in convexity structures. *Journal of Geometry*, 50(1-2):30–51.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cunha, L. and Protti, F. (2018). Closure of genomic sets: applications of graph convexity to genome rearrangement problems. *Electronic Notes in Discrete Mathematics*, 69:285–292.

Cunha, L. and Protti, F. (2019). Genome rearrangements on multigenomic models: Applications of graph convexity problems. *Journal of Computational Biology*, 26(11):1214–1222.

Dai, X., Lopez-Pintado, S., and Alzheimer's Disease Neuroimaging Initiative (2022). Tukey's depth for object data. *Journal of the American Statistical Association*, pages 1–13.

Davey, B. A. and Priestley, H. A. (2002). *Introduction to Lattices and Order, Second Edition*. Cambridge University Press.

de Araújo, P. H. M., Campêlo, M. B., Corrêa, R. C., and Labbé, M. (2019). The geodesic classification problem on graphs. In *Proceedings of the tenth Latin and American Algorithms, Graphs and Optimization Symposium, LAGOS 2019, Belo Horizonte, Brazil, June 2-7, 2019*, volume 346 of *Electronic Notes in Theoretical Computer Science*, pages 65–76. Elsevier.

de Fraysseix, H. and de Mendez, P. O. (2012). Trémaux trees and planarity. *European Journal of Combinatorics*, 33(3):279–293.

Der, R. and Lee, D. (2007). Large-margin classification in banach spaces. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 91–98, San Juan, Puerto Rico. PMLR.

Diestel, R. (2012). *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Djidjev, H. N. (2006). A linear-time algorithm for finding a maximal planar subgraph. *SIAM Journal on Discrete Mathematics*, 20(2):444–462.

Donoho, D. L. and Gasko, M. (1992). Breakdown Properties of Location Estimates Based on Halfspace Depth and Projected Outlyingness. *The Annals of Statistics*, 20(4):1803 – 1827.

Dourado, M. C., Gimbel, J. G., Kratochvíl, J., Protti, F., and Szwarcfiter, J. L. (2009). On the computation of the hull number of a graph. *Discrete Mathematics*, 309(18):5668–5674.

Dourado, M. C., Protti, F., Rautenbach, D., and Szwarcfiter, J. L. (2010). Some remarks on the geodetic number of a graph. *Discrete Mathematics*, 310(4):832–837.

Dua, D. and Graff, C. (2017). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

Duchet, P. (1987). Convexity in combinatorial structures. In *Proceedings of the 14th Winter School on Abstract Analysis*, pages 261–293. Circolo Matematico di Palermo.

Duchet, P. (1988). Convex sets in graphs, ii. minimal path convexity. *Journal of Combinatorial Theory, Series B*, 44(3):307–316.

Duchet, P. and Meyniel, H. (1983). Ensemble convexes dans les graphes i: Théorèmes de helly et de radon pour graphes et surfaces. *European Journal of Combinatorics*, 4(2):127–132.

Ellis, J. W. (1952). A general set-separation theorem. *Duke Mathematical Journal*, 19(3):417–421.

Erdos, P., Rényi, A., et al. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60.

Everett, M. G. and Seidman, S. B. (1985). The hull number of a graph. *Discrete Mathematics*, 57(3):217–223.

Farber, M. and Jamison, R. E. (1987). On local convexity in graphs. *Discrete Mathematics*, 66:231–247.

Farkas, J. (1902). Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902(124):1–27.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Fukumizu, K., Lanckriet, G., and Sriperumbudur, B. K. (2011). Learning in hilbert vs. banach spaces: A measure embedding viewpoint. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Ganter, B. and Reuter, K. (1991). Finding all closed sets: A general approach. *Order*, 8(3):283–290.

Ganter, B., Stumme, G., and Wille, R. (2005). *Formal Concept Analysis, Foundations and Applications*, volume 3626. Springer-Verlag, Berlin, Heidelberg.

Gély, A. (2005). A generic algorithm for generating closed sets of a binary relation. In *Formal Concept Analysis, Third International Conference, ICFCA 2005, Lens, France, February 14-18, 2005, Proceedings*, volume 3403 of *Lecture Notes in Computer Science*, pages 223–234. Springer.

Gilad-Bachrach, R., Navot, A., and Tishby, N. (2004). Bayes and tukey meet at the center point. In *Learning Theory*, pages 549–563, Berlin, Heidelberg. Springer.

Gottlieb, L.-A., Kontorovich, A., and Krauthgamer, R. (2014). Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759.

Graepel, T., Herbrich, R., Bollmann-Sdorra, P., and Obermayer, K. (1999). Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems 11*, pages 438–444. MIT Press.

Grätzer, G. (2011). *Lattice theory: Foundation*. Springer Science & Business Media.

Hahn, H. (1927). Über lineare gleichungssysteme in linearen räumen. *Journal für die reine und angewandte Mathematik*, 156:214–229.

Harary, F., Loukakis, E., and Tsouros, C. (1993). The geodetic number of a graph. *Mathematical and Computer Modelling*, 17(11):89–95.

Harary, F. and Nieminen, J. (1981). Convexity in graphs. *Journal of Differential Geometry*, 16(2):185–190.

He, Q. and Li, H.-L. (2011). Separation theorem and maximal margin classification for fuzzy number spaces. *2011 International Conference on Machine Learning and Cybernetics*, 1:278–281.

Hein, M. and Bousquet, O. (2003). Maximal margin classification for metric spaces. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, 2777(July 2004):72–86.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Hornik, K., Stinchcombe, M. B., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Horváth, T., Ramon, J., and Wrobel, S. (2010). Frequent subgraph mining in outerplanar graphs. *Data Mining and Knowledge Discovery*, 21(3):472–508.

Hüllermeier, E. (2011). Fuzzy sets in machine learning and data mining. *Applied Soft Computing*, 11(2):1493–1505.

Ignatov, D. I. (2015). Introduction to formal concept analysis and its applications in information retrieval and related fields. In *Communications in Computer and Information Science*, volume 505, pages 42–141.

Jain, B. J. and Obermayer, K. (2009). Structure spaces. *Journal of Machine Learning Research*, 10:2667–2714.

Jamison, R. E. (1974). *A general theory of convexity*. PhD thesis, University of Washington.

Johnson, D. and Preparata, F. (1978). The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93–107.

Kakutani, S. (1937). Ein Beweis des Satzes von M. Eidelheit über konvexe Mengen. *Proceedings of the Imperial Academy*, 13(4):93–94.

Kay, D. and Womble, E. W. (1971). Axiomatic convexity theory and relationships between the carathéodory, helly, and radon numbers. *Pacific Journal of Mathematics*, 38(2):471–485.

Kempner, Y. and Levit, V. E. (2010). Duality between quasi-concave functions and monotone linkage functions. *Discrete Mathematics*, 310(22):3211–3218.

Kempner, Y., Mirkin, B., and Muchnik, I. (1997). Monotone linkage clustering and quasi-concave set functions. *Applied Mathematics Letters*, 10(4):19–24.

Kempner, Y. and Muchnik, I. (2003). Clustering on antimatroids and convex geometries. *WSEAS Transactions on Mathematics*, 2(1):54–59.

Khardon, R. and Arias, M. (2006). The subsumption lattice and query learning. *Journal of Computer and System Sciences*, 72(1):72–94.

Knuth, D. E. (1993). *The Stanford GraphBase: A Platform for Combinatorial Computing*. Association for Computing Machinery.

Korte, B., Lovász, L., and Schrader, R. (2012). *Greedoids*, volume 4. Springer Science & Business Media.

Kubiś, W. (2002). Separation properties of convexity spaces. *Journal of Geometry*, 74(1-2):110–119.

Kuratowski, C. (1922). Sur l'opération a de l'analysis situs. *Fundamenta Mathematicae*, 3(1):182–199.

Leipert, S. (1998). *Level Planarity Testing and Embedding in Linear Time*. PhD thesis, University of Cologne.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Leskovec, J. and Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1.

Levi, F. W. (1951). On helly's theorem and the axioms of convexity. *Journal of the Indian Mathematical Society*, 15:65–76.

Lloyd, J. W. (2012). *Foundations of logic programming*. Springer Science & Business Media.

Lowen, R. (1980). Convex fuzzy sets. *Fuzzy sets and Systems*, 3(3):291–310.

Lusseau, D. (2003). The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270.

Marc, T. and Šubelj, L. (2018). Convexity in complex networks. *Network Science*, 6(2):176–203.

Minkowski, H. (1911). *Gesammelte Abhandlungen*, volume 2. BG Teubner.

Minsky, M. and Papert, S. (1987). *Perceptrons - an introduction to computational geometry*. MIT Press.

Monjardet, B. and Raderanirina, V. (2001). The duality between the anti-exchange closure operators and the path independent choice operators on a finite set. *Mathematical Social Sciences*, 41(2):131–150.

Moore, E. (1910). *Introduction to a Form of General Analysis*. Colloquium publications / American Mathematical Society. Yale University Press.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. www.graphlearning.io.

Mozharovskyi, P. (2015). *Contributions to depth-based classification and computation of the Tukey depth*. PhD thesis, University of Cologne.

Mulder, H. M. (1980). *The interval function of a graph*. Math. Centre Tracts 132. Centrum Voor Wiskunde en Informatica.

Mullat, J. E. (1976). Extremal Subsystems of Monotonic Systems I. *Avtomatica i Telemekhanika*, 5:130–139.

Newman, M. (2018). *Networks*. Oxford university press.

Nguyen, H. T. (1978). A note on the extension principle for fuzzy sets. *Journal of mathematical analysis and applications*, 64(2):369–380.

Nienhuys-Cheng, S. and de Wolf, R., editors (1997). *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer.

Ore, O. (1944). Galois connexions. *Transactions of the American Mathematical Society*, 55(3):493–513.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46.

Pelayo, I. M. (2013). *Geodesic Convexity in Graphs*. Springer New York.

Piotrowski, W. (1987). A generalization of branch weight centroids. *Applicationes Mathematicae*, 19(3-4):541–545.

Piotrowski, W. and Syslo, M. M. (1991). Some properties of graph centroids. *Annals of Operations Research*, 33(3):227–236.

Plotkin, G. D. (1970). A Note on Inductive Generalization. *Machine Intelligence 5*, 5(8):101–124.

Riesz, F. (1909). *Stetigkeitsbegriff und Abstrakte Mengenlehre*, volume 3. Tip. della R. Accademia dei Lincei; proprietà del cav. V. Salviucci.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.

Seiffarth, F., Horváth, T., and Wrobel S. (2019). Maximal closed set and half-space separations in finite closure systems. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*, volume 11906 of *Lecture Notes in Computer Science*, pages 21–37. Springer.

Seiffarth, F., Horváth, T., and Wrobel S. (2020). Maximum margin separations in finite closure systems. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part I*, volume 12457 of *Lecture Notes in Computer Science*, pages 3–18. Springer.

Seiffarth, F., Horváth, T., and Wrobel S. (2022a). Applications of the graph tukey depth. In *Proceedings of the LWDA 2022 Workshops: FGWM, KDML, FGWI-BIA, and FGIR*, CEUR Workshop Proceedings.

Seiffarth, F., Horváth, T., and Wrobel S. (2022b). A fast heuristic for computing geodesic closures in large networks. In *Discovery Science - 25th International Conference, DS 2022, Montpellier, France, October 10-12, 2022, Proceedings*, volume 13601 of *Lecture Notes in Computer Science*, pages 476–490. Springer.

Stadtländer, E., Horváth, T., and Wrobel, S. (2021). Learning weakly convex sets in metric spaces. In *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part II*, volume 12976 of *Lecture Notes in Computer Science*, pages 200–216. Springer.

Stone, M. H. (1938). Topological representations of distributive lattices and brouwerian logics. *Časopis pro pěstování matematiky a fysiky*, 67(1):1–25.

Šubelj, L. (2018). Convex skeletons of complex networks. *Journal of The Royal Society Interface*, 15(145):20180422.

Tarski, A. (1942). Introduction to logic and to the methodology of the deductive sciences. *The Modern Schoolman*, 20(1):56–56.

Thiessen, M. and Gärtner, T. (2020). Active learning on graphs with geodesically convex classes. In *KDD workshop on Mining and Learning with Graphs*.

Thiessen, M. and Gärtner, T. (2021). Active learning of convex halfspaces on graphs. *Advances in Neural Information Processing Systems*, 34:23413–23425.

Thiessen, M. and Gärtner, T. (2022). Online learning of convex sets on graphs. In *ECMLPKDD*.

Tukey, J. W. (1942). Some notes on the separation of convex sets. *Portugaliae Mathematica*, 3:95–102.

Tukey, J. W. (1975). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, 2:523–531.

van de Vel, M. (1982). Finite dimensional convex structures I: General results. *Topology and its Applications*, 14(2):201–225.

van de Vel, M. (1984). Binary convexities and distributive lattices. *Proceedings of the London Mathematical Society*, S3-48(1):1–33.

van de Vel, M. L. J. (1993). *Theory of Convex Structures, Volume 50 (North-Holland Mathematical Library)*. North Holland.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60. https://doi.org/10.1145/2641190.2641198.

Vapnik, V. and Chervonenkis, A. (1974). *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

Vashist, A., Kulikowski, C. A., and Muchnik, L. (2007). Ortholog clustering on a multipartite graph. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):17–27.

von Luxburg, U. and Bousquet, O. (2004). Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5:669–695.

Šubelj, L., Fiala, D., Ciglaric, T., and Kronegger, L. (2019). Convexity in scientific collaboration networks. *Journal of Informetrics*, 13(1):10–31.

Ward, M. (1942). The closure operators of a lattice. *Annals of Mathematics*, pages 191–196.

Witten, I. H., Frank, E., and Hall, M. A. (2011). In *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 587–605. Morgan Kaufmann, Boston, third edition edition.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.

Proof of Proposition 4.2.2 A

Proposition. *Let $k, l \in \mathbb{N}$ with $k > l \geq 2$. For all possible choices of k and l there exists a closure system that is k-Kakutani but not l-Kakutani. Moreover, for $l = 2$ and $k = 3$ there exist a l-Kakutani closure systems that is not k-Kakutani.*

Proof. To show the first claim consider the following closure system $(E, \mathcal{C})$ with $E = \{e_1, \dots, e_k\}$ and $\mathcal{C} = \{\{e_1\}, \dots \{e_k\}, E\}$. Clearly, $(E, \mathcal{C})$ is k-Kakutani, as there is only one possible choice for k pairwise disjoint closed sets, which, by definition, partitions E. In contrast, $(E, \mathcal{C})$ is not l-Kakutani because there exists no partition of E into l disjoint closed sets.

For the second claim, we consider a closure system that is 2-Kakutani but not 3-Kakutani. More precisely, let $E = \{a_1, a_2, b_1, b_2, c_1, c_2, x_1, x_2\}$. In order to define $\mathcal{C}$ over E, we simplify the notations as follows: For $\{a, b, c\}$ we write abc, and Ab denotes $A \cup \{b\}$. Let $\mathcal{S} = \{H_1, H_1^c, H_2, H_2^c, H_3, H_3^c\}$,

$$\begin{array}{ll} H_1 = Ac_1x_1 & H_1^c = Bc_2x_2 \\ H_2 = Ba_1x_1 & H_2^c = Ca_2x_2 \\ H_3 = Cb_1x_1 & H_3^c = Ab_2x_2 \end{array}$$

with $A = \{a_1, a_2\}, B = \{b_1, b_2\}$, and $C = \{c_1, c_2\}$ and let $\mathcal{C}$ be the family of all possible intersections of the sets in $\mathcal{S}$, i.e.,

$$\mathcal{C} := \{C : C = \bigcap_{X \in \mathcal{U}} X \text{ for some } \mathcal{U} \subseteq 2^{\mathcal{S}}\} \cup \{E\} \ . \tag{A.1}$$

Clearly, $(E, \mathcal{C})$ is a closure system, by definition, it is intersection closed and contains E. We first show that $(E, \mathcal{C})$ is not 3-Kakutani. Using the definition of $\mathcal{S}$ and $\mathcal{C}$, it follows that the elements in $\mathcal{S}$ are exactly the closed sets of size four in $\mathcal{C}$. Moreover, there exists no closed set of size three, as by definition of $\mathcal{S}$, the intersection of any two different elements of $\mathcal{S}$ is a set of size less than or equal to two. Considering a partition of E into three closed sets separating A, B and C, the total size of these three closed sets needs to be 8. Since a closed set cannot have size greater than 4, the cardinalities of the partitions must be either $2, 3, 3$ or $2, 2, 4$. It follows from the discussion above that the first case cannot occur. The second case is also not possible because by definition, all closed sets of size four contain one of the three sets A, B, or C, and exactly one element from the other two sets. Hence, $(E, \mathcal{C})$ is not 3-Kakutani.

In order to prove that $(E, \mathcal{C})$ is 2-Kakutani, we show that all pairs of closed sets are either half-space separable or non-disjoint. By construction, the half-spaces in $(E, \mathcal{C})$ are precisely the elements of $\mathcal{S}$ and each closed set is contained in at least one half-space. Hence, for two arbitrary closed sets X and Y, the following two cases can occur. Let $X \subseteq H_i$ for some $i \in [3]$; the case $X \subseteq H_i^c$ is analogous. Then either $Y \cap H_i = \emptyset$ or $|Y \cap H_i| > 0$. Regarding the first case, we have that H_i, H_i^c is a half-space separation of X, Y. For the second case it suffices to consider $i = 1$ for symmetry. Hence, if suffices to show that if $|Y \cap H_1| > 0$, then either X and Y are half-space separable or have a non-empty intersection. We show this by going through all possible cases for X and Y. Note that by construction, every closed set can be represented by a 3×2 binary matrix as follows: The entries in the first column of the matrix denote whether or not H_1, H_2 or H_3 are used to generate the closed set, i.e., whether they are contained in $\mathcal{U}$ using the definition of $\mathcal{C}$ in (A.1). The entries in the second column denote whether or not H_1^c, H_2^c or H_3^c are used to generate the closed set. For example, we have

$$A = H_1 \cap H_3^c \equiv \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} .$$

Thus, we can easily give a complete list of all closed sets in the closure system using their matrix representations:

🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵	🔴 🔵
H_1	a_1x_1	c_1a_2	c_1x_1	A	x_1	a_1	c_1	a_2
$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$

H_1^c	B	c_2x_2	b_1c_2	b_2x_2	b_1	b_2	c_2	x_2
$\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$

🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵
E	H_2	H_2^c	H_3	H_3^c	b_1x_1	a_1b_2	C	a_2x_2
$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$

Assuming that $X \subseteq H_1$, we have that X must be one of the closed sets from the first row denoted by the red dots; $|Y \cap H_1| > 0$ implies that Y is one of the closed sets from the first or third row denoted by the blue dots. Considering all possible combinations of closed sets X and Y Table A.1 below shows that each X (red dots) can either be separated from each Y (blue dots) by half-spaces (denoted by entries $1, 2$, or 3 depending on the

	a_1x_1	c_1a_2	c_1x_1	A	x_1	a_1	c_1	a_2
a_1x_1	$\times$	2	$\times$	$\times$	$\times$	$\times$	2	2
c_1a_2	2	$\times$	$\times$	$\times$	2	2	$\times$	$\times$
c_1x_1	$\times$	$\times$	$\times$	3	$\times$	3	$\times$	3
A	$\times$	$\times$	3	$\times$	3	$\times$	3	$\times$
x_1	$\times$	2	$\times$	3	$\times$	3	2	$2,3$
a_1	$\times$	2	3	$\times$	3	$\times$	2	2
c_1	2	$\times$	$\times$	3	2	2	$\times$	3
a_2	2	$\times$	3	$\times$	$2,3$	2	3	$\times$
b_1x_1	$\times$	2	$\times$	3	$\times$	3	2	$2,3$
a_1b_2	$\times$	2	3	$\times$	3	$\times$	3	2
C	2	$\times$	$\times$	3	2	3	$\times$	3
a_2x_2	2	$\times$	3	$\times$	$2,3$	2	3	$\times$

Table A.1: Separability of the closed sets defined in Proposition 4.2.2. The closed sets either intersect ($\times$) or are half-space separable regarding some half-spaces H_i, H_i^c for $i \in \{1, 2, 3\}$.

half-space separation H_i, H_i^c) or $X \cap Y \neq \emptyset$ (denoted by the entry $\times$). Closed sets of size four are omitted as they obviously fulfill the claim above. Hence, the above defined closure system is 2-Kakutani but not 3-Kakutani. $\square$