Competitive Skyline Cliques

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Computer Science and Engineering by Research

by

Jayitha Cherapanamjeri
20171401
jayitha.c@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
January 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Competitive Skyline Cliques" by Jayitha Cherapanamjeri, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Kamalakar Karlapalem

To mom

Acknowledgments

I thank my advisor, Prof. Kamalakar Karlapalem, for his limitless patience. I attribute my deep appreciation of good ideas to Prof. Kamal. But even more so, I thank Prof. Kamal for teaching me that good research is more than just a good idea; it is the work that is put into its realization. Prof. Kamal taught me not to judge, to not overthink, to be okay with making mistakes and to never be scared.

I thank the professors at IIIT Hyderabad for all the effort they put into enforcing me with the right ideas. I thank Prof. Lini Thomas for helping me navigate the technicalities of problem-solving. I thank Prof. P. Krishna Reddy for his wonderful courses, specifically the course titled "Advances in Database Systems". I thank Prof. Vikram Pudi for granting me the time I wanted to do my own thing. I thank Prof. Venkateshwarlu for initiating my fondness for numbers and statistics.

I thank my quality lot of friends at IIIT Hyderabad for making my time here so memorable. Mahathi, of all the things I take from here, you are my favourite. Sumukh, I thank the divine intervention responsible for our meeting and the unbelievable standard you set for friendship. I thank Vijayraj for knowing the right things to say and for our many shared interests. I thank Shyam for being the only voice of reason I entertain. I thank Pooja, we lived 20ft apart for two years before we met, and all it took was one exam, one interview, a 600 km displacement and two rooms 5ft apart.

I thank my family. I have witnessed first-hand what a ragtag group of women can do: I thank my Grandmother for rebelling and my Aunt for persisting. I thank my Mother for thriving and for her resilience. Finally, to my Brother, thank you for being an inexcusably tangible artefact of what our shared gene pool can achieve; few are as lucky as I am.

Abstract

Decision-making is hard when presented with a large set of similar options that insignificantly trade-off amongst a range of attributes. This phenomenon is encountered within the skyline set because no skyline point is strictly better than any other skyline point, and therefore, every skyline point can excel ever so slightly in *some* subspace of attributes. The objective of this work is to determine the set of all sets of skyline points that are hard to choose from and hence all points in a set *compete* for attention from the same *type* of consumer (such sets are called *competitive skyline cliques*)

In this work, two skyline points are defined to be *competitive* if the differences between the two points across each attribute are bounded by specified thresholds. We introduce *maximal competitive skyline cliques* (MCSCs) – maximal sets of mutually competitive skyline points and provide algorithms that enumerate all MCSCs. While the problem of enumerating all MCSCs is structurally similar to the NP-Hard problem of enumerating all maximal cliques in a graph, because of properties exhibited by our competitiveness metric, we show that enumerating all MCSCs can be performed efficiently in polynomial time. Furthermore, in 2D space, we provide an optimal linear-time sweepline algorithm. We also provide a bounded approximation for MCSCs that are easier to enumerate. Our extensive experiments using both synthetic and real (UCars, FIFA22 and Pokémon) datasets demonstrate the efficacy of MCSCs and the efficiency of our algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overchoice Effect

In market research, it is generally believed that consumers prefer having a large set of options (called *choice sets*) to choose from. However, experiments conducted by [Iyengar and Lepper, 2001] showed that large choice sets do not always lead to higher consumer satisfaction. In fact, in their experiments, they found that consumers who chose from large choice sets were often less satisfied than consumers who chose from smaller ones. This phenomenon is called the *overchoice* or the *choice overload* effect. [Greifeneder et al., 2010] credited this phenomenon to *increase choice complexity*. The choice complexity of a choice set is a measure of how difficult picking within a choice set is. The choice complexity is characterised by three properties of the choice set.

1. Choice set cardinality: The number of options in the choice set.
2. Amount of information: The number of attributes that are recorded for each option in the set. If there are a large number of attributes, the choice set is termed to suffer from *too-much-information*.
3. Similarity between options: If two options are practically indiscernible, then it is hard to pick between them. This effect gets increasingly complex when there are a large number of similar options.

A choice set is said to have a high choice complexity if it is large, has many attributes and has large clusters of highly similar options. The overchoice effect that occurs as a result of increased choice complexity is attributed to three factors that add to decreased consumer satisfaction.

1. A consumer's final choice may be suboptimal. If one product is picked, all other products, possibly including the optimal, are eliminated from choice.
2. Consumers may need a long time to search through the choice set. This adversely affects consumer satisfaction.
3. Irrespective of whether the final choice is optimal, consumers may be left feeling uncertain about their choice.

The objective of this work is to identify subsets of a given choice set where the overchoice effect is most likely to occur. It is important to note that we do not provide any guidance on which choice to make; that is completely up to the consumer.

1.2 The Skyline Operator

The skyline operator[Borzsony et al., 2001] is a popular solution to the multi-objective optimisation-problem. The skyline operator is popular because it is fairly intuitive and easy to implement: Given a dataset of products/options/points D, the skyline operator eliminates strictly suboptimal points. Consider the constructed hotels dataset shown in Fig. 1.1. The dataset consists of ten hotels with two attributes – (1) the cost per night (Cost) and (2) the distance of the hotel from the beach (Distance). Hotels that are cheaper and closer to the beach are preferred. In this dataset, the skyline operator eliminates hotels p_9 and p_{10} because there exist strictly superior hotels. For instance, hotel p_7 is both cheaper and closer to the beach than hotels p_9 and p_{10}.

Formally, given a d-dimensional dataset D and a preference over the set of dimensions (`MIN`, `MAX`), the skyline set ($\mathcal{S}$) is the set of all points in D that *are not dominated* by other points in D. A d-dimensional point p *dominates* another point q ($p \prec q$) if p is *at least as good as* q in all dimensions ($\forall_d\, p[d] \preceq_d q[a]$) and p is *strictly better* than q in at least one dimension ($\exists_d\, p[d] \prec q[d]$).

$$p \prec q \iff (\forall_d\, p[d] \preceq q[d]) \wedge (\exists_d\, p[d] \prec q[d])$$

Where $p[d] \preceq q[d]$ denotes that the value of $p[d]$ is *better than or equal to* $q[d]$ and $p[d] \prec q[d]$ denotes that the value of $p[d]$ is *strictly better* than $q[d]$. A point p is a skyline point if it is not dominated by any other point in the dataset. The skyline set is the set of all skyline points.

$$\mathcal{S} = \{p \mid p \in D, \neg\exists_{q \in D}\, q \prec p\}$$

The skyline operator is a popular precursor in decision-making systems. Restricting the search space to the skyline set provides many advantages – (1) The skyline set is a subset of the dataset, the choice set is smaller (reduced choice complexity) and (2) the consumer will not pick a non-skyline point and hence does not make a suboptimal choice. If a consumer picks a skyline point, then they are guaranteed to know that there exists no strictly superior alternative in the dataset.

While the ordinary skyline operator attempts to tackle the choice set cardinality, variations of the skyline query can address the problem of too-much-information. For instance,

1. Subspace Skylines[Tao, 2006] – Frequently, the consumer is only interested in a subset of the attributes and not all of them. The skyline is then computed only for a specified subspace of interest. For instance, the consumer may not be interested in being close to the beach; then the consumer picks

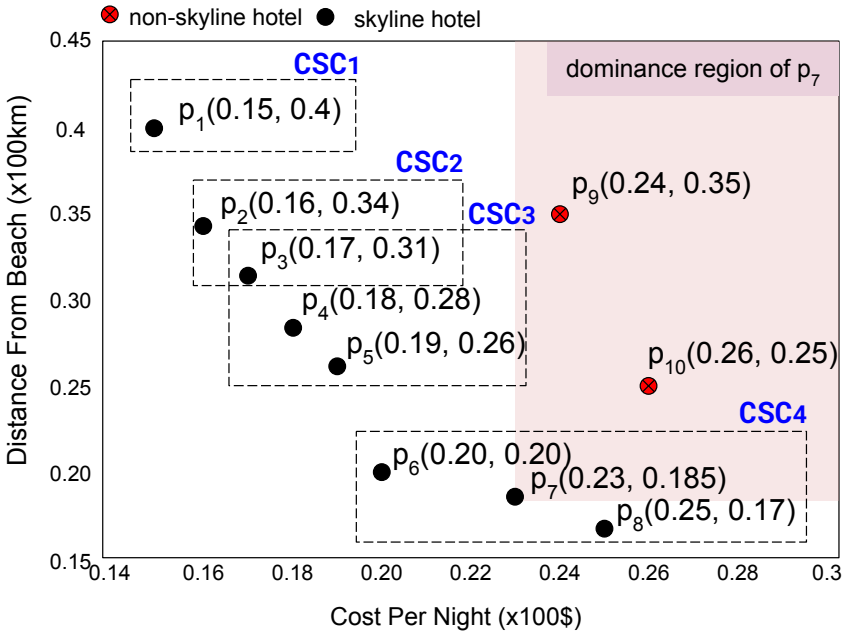


Figure 1.1: Sample hotels dataset. The skyline set is computed by minimising accross both Price and Distance. The red region depicts the dominance region of hotel p_7. Four competitive skyline cliques (CSC_1–CSC_4) have been identified (in blue).

the cheapest hotel. Reducing the dimensionality of the space of the dataset also typically reduces the size of the skyline set[Borzsony et al., 2001].

2. Constrained Skylines[Papadias et al., 2003a] – The consumer will likely have hard constraints on some attributes. For instance, the consumer may have a budget constraint to satisfy when picking a hotel. If the budget is 20$, then hotels p_7 and p_8 are eliminated, and the consumer can pick the best hotel from those remaining, p_1–p_6 (likely hotel p_6)

Despite the existence of many variants of the skyline operator, picking between skyline points can be challenging. No skyline point dominates any other skyline point; every skyline point excels in *some* subspace of attributes. Therefore, when making a choice, the consumer has to rationalize over the presented tradeoffs. These tradeoffs can get complicated when the number of attributes and options are large and when these quantified tradeoffs are practically insignificant. This work introduces a new skyline variant that generates sets of skyline points that are similar and hard to choose from.

1.3 Motivation and Contributions

Picking amongst hotels in the skyline set can be challenging. When comparing any two skyline points, the consumer has to decide between the presented tradeoffs. Consider hotels p_1 and p_8 – hotel p_1 is the cheapest hotel in the dataset, and hotel p_8 is closest to the beach. These hotels are polarising;

they differ prominently by a 10$ cost difference and a 23km distance difference. On the other hand, consider hotels p_7 and p_8 – they are both expensive and close to the beach. They differ insignificantly by a 2$ cost and 0.5km distance difference. Picking between hotels p_7 and p_8 is harder than picking between hotels p_1 and p_8. Making this choice becomes significantly harder when the tourist is presented with a large set of similar hotels and more attributes [Iyengar and Lepper, 2001, Greifeneder et al., 2010, Scheibehenne, 2008].

CSC	Car ID	Name	Year	Selling Price ($)	Driven (km)	Fuel	Transmission	# Owners	Mileage (kmpl)	Engine (CC)	Max Power (bhp)	Seats
1	1	Audi A4 35 TDI Premium Plus	2018	38600	15000	Diesel	Automatic	1	18.25	1968	187.74	5
	2	Audi Q3 2.0 TDI Quattro Premium Plus	2017	37600	22000	Diesel	Automatic	1	15.73	1968	174.33	5
	3	BMW 5 Series 520d Luxury Line	2016	38600	12000	Diesel	Automatic	1	18.12	1995	190	5
2	4	Ford Figo 1.2P Ambiente MT	2011	3200	25000	Petrol	Manual	1	18.16	1196	86.8	5
	5	Ford Figo Petrol EXI	2011	3000	15000	Petrol	Manual	1	15.6	1196	70	5
	6	Hyundai i10 Era	2010	2800	20000	Petrol	Manual	1	19.81	1086	68.05	5
	7	Maruti Ritz VXi	2011	3000	18500	Petrol	Manual	1	18.5	1197	85.8	5
	8	Maruti Swift VXI	2011	3700	20000	Petrol	Manual	2	18.6	1197	85.8	5
3	9	Tata Winger Deluxe - Flat Roof (Non-AC)	2010	3100	50000	Diesel	Manual	1	10.71	1948	90	14

Figure 1.2: Sample CSCs of UCars dataset

As another example, consider a sample from the skyline set of a Used Cars (UCars) dataset shown in Fig. 1.2. This sample consists of nine used cars with eleven attributes. Picking between the first and ninth cars is easy – these cars clearly target different audiences. The first car, an Audi, is an expensive, powerful, high-end car that seats five people. Whereas, the last car, a Tata MPV (**M**ulti**p**urpose utility **V**ehicle) is a much bigger car that seats fourteen people. On the other hand, picking between the sixth (a Hyundai i10) and the eighth (a Maruti Swift) cars is clearly much harder — where one (the Hyundai i10) is cheaper and has better mileage, the other (the Maruti Swift) was manufactured more recently and is more powerful. Furthermore, these differences are insignificant; how much should the consumer value a 1.2 kmpl mileage difference? The Hyundai i10 and the Maruti Swift target the same set of consumers.

In the hotels example, hotels p_7 and p_8 are similar and *compete* for attention from the same kinds of consumers, i.e. those that prefer hotels closest to the beach. In this work, we define two skyline points to be *competitive* if the differences between the attributes of the two points are at most ε, a user-specified

threshold. For instance, if $\varepsilon = 0.05$, then hotels p_7 and p_8 are competitive and hotels p_1 and p_8 are not.

$$
\begin{aligned}
|p_1.\texttt{price} - p_8.\texttt{price}| &= |0.15 - 0.25| &&= 0.1 > \varepsilon(0.05) \\
|p_1.\texttt{distance} - p_8.\texttt{distance}| &= |0.400 - 0.17| &&= 0.23 > \varepsilon(0.05) \\
|p_7.\texttt{price} - p_8.\texttt{price}| &= |0.23 - 0.25| &&= 0.02 \leq \varepsilon(0.05) \\
|p_7.\texttt{distance} - p_8.\texttt{distance}| &= |0.185 - 0.17| &&= 0.015 \leq \varepsilon(0.05)
\end{aligned}
$$

The objective of our work is to generate sets of mutually competitive skyline points (called *competitive skyline cliques*). For instance, in Fig. 1.1, four competitive skyline cliques have been identified (CSC_1–CSC_4). Each of these cliques is a set of similar hotels that are hard to compare and select from.

Contributions

The contributions of this thesis are summarised below.

1. We formally define competitive skyline cliques (CSC) and maximal competitive skyline cliques (MCSC) (Chapter 2) and adapt an output-sensitive maximal clique enumeration algorithm to enumerate all MCSCs (Chapter 4).

2. We provide optimisationsthat improve the implementation efficiency of the proposed algorithm (Chapter 5).

3. We derive theoretical bounds on the number of CSCs ($\mathcal{O}(2^{|\mathcal{S}|})$) and MCSCs ($\mathcal{O}(|\mathcal{S}|^d)$) (Chapter 6).

4. Due to properties exhibited by the 2D skyline set, we provide an optimal bound on the number of 2D MCSCs and an optimal algorithm that enumerates all 2D MCSCs (Chapter 8).

5. Since enumerating all MCSCs can be computationally expensive, we provide an approximation for MCSCs (AMCSC) that are fewer in number and easier to enumerate (Chapter 9).

6. Finally, we experimentally analyse MCSCs of real datasets (UCars, FIFA22 and Pokémon) and the effect of the dimensionality (d), cardinality(n) and threshold (ε) on MCSCs of synthetically generated datasets (Chapter 7)

Chapter 2

Competitive Skyline Cliques

Notation and Assumptions. In this thesis, we use $D = \{p_1, p_2, \ldots, p_n\}$ to denote a d-dimensional n-sized dataset, $\mathcal{A} = \{a_1, a_2, \ldots, a_d\}$ to denote the set of d attributes or dimensions and $\mathcal{S}$ to denote the skyline set. Unless otherwise specified, points p and q denote d-dimensional data points and $p[a]$ or $p.a$ denote the value of attribute a ($a \in \mathcal{A}$) of point p. Additionally, without loss of generality, we assume that the domain of every attribute is $[0, 1]$ and that smaller values of attributes are better, i.e. the skyline set is obtained by minimising across all attributes. The notations used in this thesis are summarised in Table 2.1

Table 2.1: Summary of notations

Notation	Definition
p, q	skyline points
$\mathcal{S}$	skyline set
$\mathbf{s}, \mathbf{s}'$	subsets of skyline set
$\mathbf{c}$	competitive skyline clique (abbrev. CSC)
$\mathcal{C}$	set of all competitive skyline cliques
$\mathbf{m}$	maximal competitive skyline clique (abbrev. MCSC)
$\mathcal{M}$	set of all maximal competitive skyline cliques
ε	specified threshold in range $[0, 1]$
$comp(p, q)$	p and q are competitive ($\|\|p - q\|\|_\infty \leq \varepsilon$)
$comp(p, \mathbf{s})$	p is competitive with *all* points in $\mathbf{s}$ ($competitors(p, \mathbf{s}) = \mathbf{s}$)
$competitors(p, \mathbf{s})$	set of points in $\mathbf{s}$ that are competitive with p
$competitors(\mathbf{s}', \mathbf{s})$	set of points in $\mathbf{s}$ that competitive with *all* points in $\mathbf{s}'$
$parcomp(p, \mathbf{s})$	p is competitive with *at least one* point in $\mathbf{s}$
$\neg parcomp(p, \mathbf{s})$	p is *not* competitive with *any* point in $\mathbf{s}$ ($competitors(p, \mathbf{s}) = \emptyset$)

Definition 2.1

Two skyline points p and q are defined to be **competitive** ($comp(p, q)$) if the Chebyshev distance (or infinity norm distance) between points p and q is at most ε, a specified threshold in the range $[0, 1]$.

$$comp(p, q) \iff ||p - q||_\infty \leq \varepsilon$$

The Chebyshev distance is calculated as follows: $\max_{a \in \mathcal{A}} |p[a] - q[a]|$. Notation $\neg comp(p, q)$ is used to denote that skyline points p and q are not competitive.

The use of the Chebyshev distance metric implies that for two skyline points p and q to be competitive, the difference between points p and q across all attributes must be bound by ε. While in this document, a single value of ε bounds the difference across all attributes, all definitions, properties and algorithms presented as part of this thesis can be extended for when different thresholds are provided for each attribute. Given the competitiveness metric, we can construct a graph that represents *competitive* relationships between skyline points (called **competitiveness graph** or **comp graph** for short). The comp graph is the graph $\mathbf{G}[\mathbf{V}, \mathbf{E}]$ where the vertices $\mathbf{V}$ and the edges $\mathbf{E}$ are:

$$\mathbf{V} = \mathcal{S}$$
$$\mathbf{E} = \{(u, v) \mid u, v \in \mathcal{S} \text{ and } comp(u, v)\}$$

An edge exists between two vertices if and only if the corresponding skyline points are competitive. The comp graph of the hotels dataset when $\varepsilon = 0.05$ is shown in Fig. 2.1. The comp graph is expected to be sparse since ε, the specified threshold, is assumed to be small.

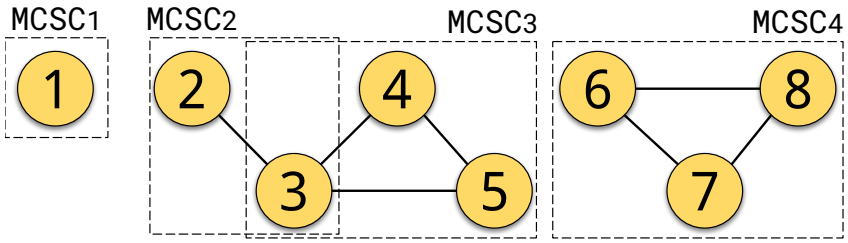


Figure 2.1: competitiveness graph of hotels dataset when $\varepsilon = 0.05$

Given a skyline point p and a subset of skyline set $\mathbf{s}$, $competitors(p, \mathbf{s})$ is the set of all points in $\mathbf{s}$ that are competitive with p. Consequently, $competitors(p, \mathcal{S})$ is the set of all skyline points that are competitive with skyline point p (including p). From this point forward in the document, when we refer to a point p, we implicitly mean we are referring to a *skyline* point p.

Example. In the hotels dataset, when $\varepsilon = 0.05$, hotels p_2 and p_3 are competitive ($||p_2 - p_3||_\infty = 0.02 \leq 0.05 = \varepsilon$). For the rest of this paper, we assume $\varepsilon = 0.05$ for the hotels dataset unless otherwise specified. Hotels p_1 and p_8 are not competitive, denoted by $\neg comp(p_1, p_8)$. The set of all *competitors*

of hotel p_3 is

$$competitors(p_3, \mathcal{S}) = \{p_2, p_3, p_4, p_5\}$$

In this document, specifically for the hotels dataset, we use a concise notation to represent sets: set $\{p_6, p_7, p_8\}$ is represented as 678. Therefore, $competitors(p_3, \mathcal{S}) = 2345$ and $competitors(p_2, 2345) = 23$.

We extend notations $comp(,)$ and $competitors(,)$ to sets — given a skyline point p and subsets of the skyline set $\mathbf{s}$ and $\mathbf{s}'$, $comp(p, \mathbf{s})$ denotes that point p is competitive with *all* points in set $\mathbf{s}$ and $competitors(\mathbf{s}', \mathbf{s})$ is the set of all points in $\mathbf{s}$ that are competitive with *all* points in set $\mathbf{s}'$. Furthermore, we use notation $parcomp(p, \mathbf{s})$ to denote that skyline point p is competitive with *at least one point* in $\mathbf{s}$, a subset of skyline set. Consequently, $\neg parcomp(p, \mathbf{s})$ denotes that point p is not competitive with any point in set $\mathbf{s}$.

Example. Hotel p_3 is competitive with all hotels in set 2345 – $comp(p_3, 2345)$. Hotel p_3 is the only hotel that is competitive with both hotels p_2 and p_4 from set 2345 – $competitors(24, 2345) = 3$. Hotel p_2 is competitive with at least one hotel in set 2345 – $parcomp(p_2, 2345)$ whereas hotel p_1 is not competitive with any hotel in 2345 – $\neg parcomp(p_1, 2345)$.

Definition 2.2

Competitive skyline cliques (CSCs) are sets of mutually competitive skyline points. Formally, a subset of the skyline set ($\mathbf{c} \subseteq \mathcal{S}$) is defined to be **competitive** (also called a **competitive skyline clique** or CSC for short) if every point in $\mathbf{c}$ is competitive with every other point in $\mathbf{c}$ Furthermore, a CSC $\mathbf{m}$ is defined to be **maximal** (also called a **maximal competitive skyline cliques** or MCSC for short) if no other skyline point is competitive with all points in $\mathbf{m}$
$\mathcal{C}$ and $\mathcal{M}$ denote the set of all CSCs and MCSCs respectively.

A clique in a graph is a subset of vertices such that every pair of vertices in the clique is connected by an edge. A clique is *maximal* if it is not a subset of any other clique. Equivalently, each CSC is a clique in the comp graph and, each MCSC is a maximal clique in the comp graph. Note that CSCs and MCSCs are not exclusive, i.e. they can overlap. For instance MCSCs 23 and 345 overlap.

Example. The following are all valid CSCs of the hotels dataset: 6, 68, 678. The set of all CSC of the hotels dataset (denoted by $\mathcal{C}$) is

$$\mathcal{C} = \{1, 2, 3, 4, 5, 23, 34, 35, 45, 345, 6, 7, 8, 67, 68, 78, 678\}$$

CSCs can be redundant, for instance CSCs 68 and 678 practically target the same consumers but CSC 678 correctly quantifies the competition for the target consumers. MCSCs eliminate these redundant CSCs. The set of all MCSCs of the hotels dataset (denoted by $\mathcal{M}$) is

$$\mathcal{M} = \{1, 23, 345, 678\}$$

MCSCs of the hotels dataset are depicted in Fig. 2.1 as equivalent maximal cliques.

We detail two interesting properties of CSCs — (1) the Apriori property (Property 2.1): all subsets of CSCs are CSCs and (2) the maximality condition (Property 2.2): a CSC $\mathbf{m}$ is maximal if and only if the set of competitors of $\mathbf{m}$ in the skyline set is $\mathbf{m}$.

Property 2.1 (Apriori Property)**.** If $\mathbf{c}$ is a CSC then all subsets of $\mathbf{c}$ are CSCs

Proof. We use a proof by contradiction. Let $\mathbf{c}$ be a CSC and $\mathbf{c}'$ be a subset of $\mathbf{c}$ such that $\mathbf{c}'$ is not competitive.

$$\begin{aligned} \mathbf{c}' \text{ is not competitive} &\iff \exists_{p,q \in \mathbf{c}'} \neg comp(p,q) \\ &\iff p,q \in \mathbf{c} \ \wedge \neg comp(p,q) \\ &\iff \mathbf{c} \text{ is not competitive} \qquad \text{(contradiction)} \end{aligned}$$

□

The Apriori property only provides a necessary condition (single direction implication) when, in fact, CSCs also satisfy the sufficiency condition (double direction implication). If all subsets of a set of skyline points $\mathbf{c}$ are competitive, then $\mathbf{c}$ is competitive.

Property 2.2 (Maximality Condition)**.** A CSC $\mathbf{m}$ is maximal if and only if

$$\mathbf{m} = competitors(\mathbf{m}, \mathcal{S})$$

Proof. We use a proof by contradiction. Let $\mathbf{m}$ be a MCSC and let $\mathbf{m}$ and $competitors(\mathbf{m}, \mathcal{S})$ be unequal. There are two cases: there exists a skyline point that is in $\mathbf{m}$ but not in $competitors(\mathbf{m}, \mathcal{S})$ or there exists a skyline point that is in $competitors(\mathbf{m}, \mathcal{S})$ but not in $\mathbf{m}$. Let p be a skyline point such that

- $p \in \mathbf{m}; p \notin competitors(\mathbf{m}, \mathcal{S})$

$$\begin{aligned} p \notin competitors(\mathbf{m}, \mathcal{S}) &\iff \exists_{q \in \mathbf{m}} \neg comp(p,q) \\ &\iff p,q \in \mathbf{m} \ \wedge \neg comp(p,q) \\ &\iff \mathbf{m} \text{ is not competitive} \qquad \text{(contradiction)} \end{aligned}$$

- $p \notin \mathbf{m}; p \in competitors(\mathbf{m}, \mathcal{S})$

$$\begin{aligned} p \in competitors(\mathbf{m}, \mathcal{S}) \wedge p \notin \mathbf{m} &\iff \exists p \in \mathcal{S} \setminus \mathbf{m} \ comp(p, \mathbf{m}) \\ &\iff \mathbf{m} \text{ is not maximal} \qquad \text{(contradiction)} \end{aligned}$$

□

These properties are utilised when enumerating all MCSCs. We restrict our attention to MCSCs because CSCs can be (1) redundant and (2) large in number (Chapter 6).

Chapter 3

Related Work

3.1 Skyline Query Processing

In this work, we assume that the skyline set has been computed. There exist several efficient computation algorithms — BNL [Borzsony et al., 2001], D&C [Borzsony et al., 2001], bitmap [Tan et al., 2001], index [Tan et al., 2001], NN [Kossmann et al., 2002], BBS [Papadias et al., 2003a], and SFS [Chomicki et al., 2003] to name a few.

3.2 Frequent Itemset Mining

The problems of enumerating CSCs and MCSCs are algorithmically similar to the problem of mining frequent (FIM) and maximal (and closed) frequent itemsets (MFIM), respectively. Let $\mathcal{I}$ be a universal set of items, and $\mathcal{T}$ be a set of transactions where each transaction t is a subset of $\mathcal{I}$ (called an itemset), then an itemset I is defined to be *frequent* if the *support* of I is at least equal to min_sup, a specified threshold. The support of an itemset I (denoted by $support(I)$) is the number of transactions in $\mathcal{T}$ that contain itemset I.

$$support(I) = |\{t \in \mathcal{T} \mid I \subseteq t\}|$$

A frequent itemset I is defined to be *maximal* if every strict superset of I is not frequent. Frequent itemsets sets also satisfy the Apriori property:

Property 3.1 (Apriori Property)**.** If an itemset I is not frequent, all supersets of I are not frequent

The Apriori property is used by many algorithms to compute frequent itemsets that can consequently enumerate all CSCs: For instance, both the Apriori [Agrawal et al., 1994] and the Eclat [Zaki et al., 1997] algorithms can be used to enumerate all CSCs. By definition, maximal frequent itemsets are analogous to MCSCs. Most MFIM algorithms such as MAFIA [Burdick et al., 2005], GenMax [Gouda and Zaki, 2005], and LCMax [Uno et al., 2003] can be used to enumerate all MCSCs. The analog of a "transaction" is a set of skyline points that are contained in a box of size at most ε. We opted for a sparse maximal clique

enumeration since FIM algorithms are optimised to reduce the number of scans over the transaction database and not the universal set of items. Furthermore, most general-purpose FIM algorithms are not optimised for sparse patterns.

3.3 Maximal Clique Enumeration

Any maximal clique enumeration (MCE) algorithm can be adapted to enumerate all MCSCs. Furthermore, we can utilise existing MCE algorithms that are optimized for sparse graphs. For instance the classical Bron-Kerbosch [Bron and Kerbosch, 1973] or the matrix multiplication algorithm presented by [Makino and Uno, 2004] (sparse graphs). Furthermore, MCSCs can overlap significantly in certain cases, if the objective is to generate near mutually exclusive cliques of competitive products, then methods proposed by [Wang et al., 2013] that generates a subset of maximal cliques that maximise the overlap of non-represented maximal cliques or by [Sade and Cohen, 2020] that performs diverse maximal clique enumeration can be used.

3.4 Clustering

This work can be construed as a form of overlapping density-based clustering over the skyline set. But this work differs from clustering in the objective and consequently in the intention behind the choosing of threshold ε. The objective of clustering is to group points into clusters such that points in a cluster are more similar to points in the same cluster when compared to points in a different cluster. Clustering aims at identifying inherent groups present in the dataset; thresholds are picked to fit the dataset. Whereas in this work, the results returned are indications of how the dataset fits the provided thresholds and not the other way around. Thresholds are picked independently of the distribution of the data. Furthermore, clustering sacrifices granularity for better summarization; clustering algorithms can group two non-competitive products together if they are densely connected.

Chapter 4

Enumerating Competitive Skyline Cliques

Fig. 4.1 illustrates the flowchart that enumerates all MCSCs of a given dataset. The skyline of the dataset is computed using some popular skyline computation algorithm (in this work, we opted to use the Branch and Bound Skyline Computation algorithm (BBS) [Papadias et al., 2003b]). Then we attempt to partition the skyline set into disjoint and mutually exclusive sets of skyline points such that no point in one partition set is competitive with any other point in any other partition set (Section 5.1). Finally, we enumerate all MCSCs of each partition using the Output-Sentitive MCSC Enumeration (OMCE) algorithm that we present in this section.

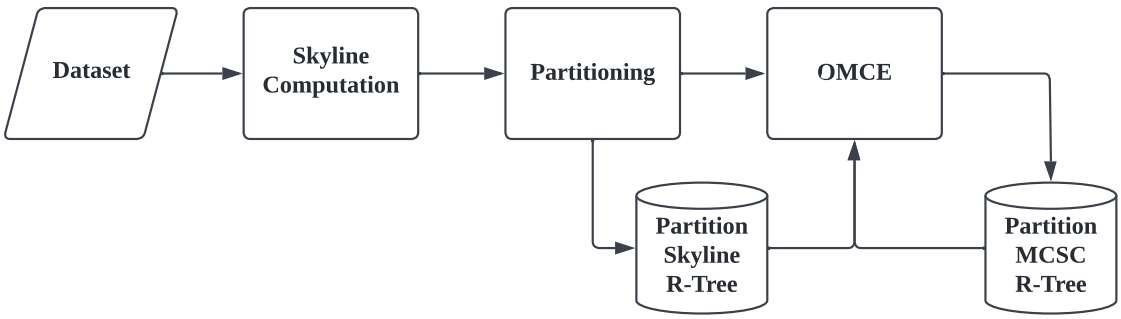


Figure 4.1: MCSC Enumeration Flowchart

The algorithm we develop in this work is inspired by the output-sensitive maximal independent set enumeration algorithm proposed by [Tsukiyama et al., 1977].

4.1 Overview

The algorithm by [Tsukiyama et al., 1977] computes all maximal cliques of graph G from all maximal cliques of graph $G \setminus v$, i.e. subgraph of G with vertex v and all its incident edges removed. We use a similar paradigm and generate all MCSCs of a set of skyline points $\mathbf{S}$ from all MCSCs of set $\mathbf{S} \setminus \{p\}$

where p is a skyline point in $\mathbf{S}$. Let $\mathbf{m}$ be a MCSC of $\mathbf{S} \setminus \{p\}$, then one of the three following cases (illustrated in Fig. 4.2) occur:

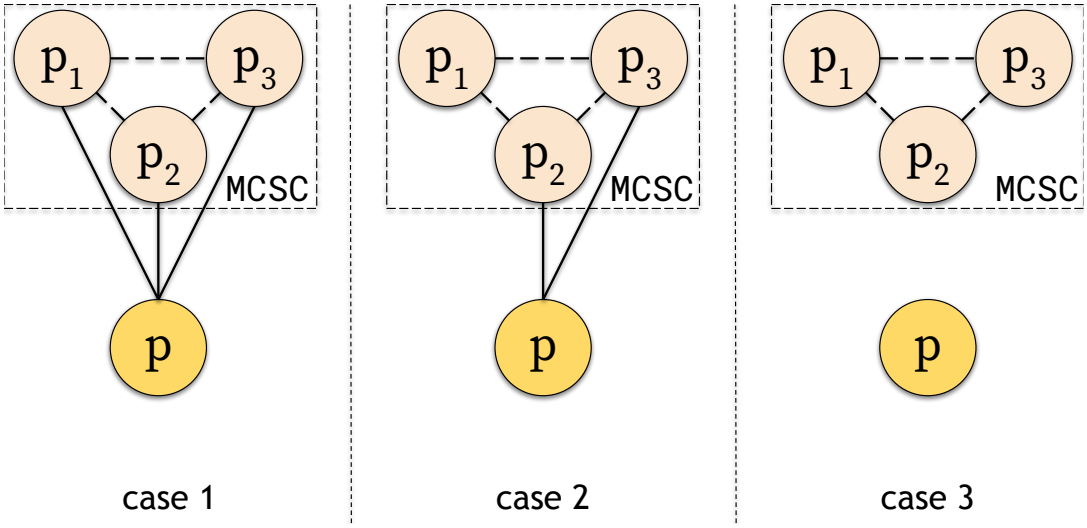


Figure 4.2: Output-sensitive clique enumeration cases

case 1 $comp(p, \mathbf{m})$: If p is competitive with *every point* in $\mathbf{m}$, then $\mathbf{m} \cup \{p\}$ is a MCSC of $\mathbf{S}$ ($\mathbf{m}$ is updated to include point p).

case 2 $parcomp(p, \mathbf{m})$: If p is competitive with *at least one point* in $\mathbf{m}$, then (1) $\mathbf{m}$ is a MCSC of $\mathbf{S}$ and (2) $competitors(p, \mathbf{m}) \cup \{p\}$ is a CSC and a **potential** MCSC of $\mathbf{S}$. $competitors(p, \mathbf{m}) \cup \{p\}$ could be subsumed by some other MCSC of $\mathbf{S}$. To verify maximality, we use Property 2.2.

case 3 $\neg parcomp(p, \mathbf{cc})$: If p is *not competitive with any point* in $\mathbf{m}$ then $\mathbf{m}$ is a MCSC of $\mathbf{S}$.

Additionally, note that if point p is not competitive with any point in $\mathbf{S} \setminus \{p\}$, then set $\{p\}$ is a MCSC of $\mathbf{S}$. There has been a huge body of work dedicated to the computation of the skyline set (Chapter 3); therefore, in our work, we assume that the skyline set is precomputed and provided as input to our algorithm.

4.2 Output-Sensitive Maximal Competitive Skyline Clique Enumeration

The output-sensitive maximal clique enumeration algorithm (Algorithm 1) generates all MCSC of set $\mathbf{S}$ from those of set $\mathbf{S} \setminus \{p\}$. The algorithm iterates through the skyline set in line 3. For each skyline point, line 5 iterates through MCSCs identified by processing skyline points until point $p_{(i-1)}$;

all MCSCs identified before processing p_i only contain points from the set $\{p_1, \ldots, p_{(i-1)}\}$. If p_i is competitive with CSC $\mathbf{m}$, then clique $\mathbf{m}$ is expanded to include point p_i (case 1). No new set is generated. Else if p_i is competitive with some points in $\mathbf{m}$, then a candidate is generated by including p_i and the points in $\mathbf{m}$ that are competitive with p_i (case 2). This candidate CSC (called *cand*) may not be maximal within set $\{p_1, \ldots, p_i\}$. Therefore, we perform a maximality check based on Property 2.2 in line 13. Note that it is possible for the same MCSC to be generated multiple times in lines 13-14 from different CSCs. Therefore, duplicates need to be eliminated. They can be eliminated by using a hash table to store the set of $candidates$ or by sorting the candidates and removing duplicates. Finally, all remaining candidates are maximal and can be included in $\mathcal{M}$. If point p_i is not competitive with any points in set $\{p_1, \ldots, p_i\}$, then singleton CSC $\{p_i\}$ is maximal and is added to $\mathcal{M}$ in lines 18-19.

Algorithm 1 **O**utput-Sensitive **M**aximal **C**ompetitive Skyline Clique **E**numeration Algorithm (OMCE)

1: **Input:** $\mathcal{S} = \{p_1, p_2, \ldots, p_m\}, \varepsilon \in [0, 1]$
2: $\mathcal{M} \longleftarrow \emptyset$
3: **for** each $p_i \in \mathcal{S}$ **do**
4: $\quad candidates \longleftarrow \emptyset$
5: $\quad$ **for** each $\mathbf{m} \in \mathcal{M}$ **do**
6: $\quad\quad$ // ▷ *case 1*
7: $\quad\quad$ **if** $comp(p_i, \mathbf{m})$ **then**
8: $\quad\quad\quad \mathbf{m}.insert(p_i)$
9: $\quad\quad$ // ▷ *case 2*
10: $\quad\quad$ **else if** $parcomp(p_i, \mathbf{m})$ **then**
11: $\quad\quad\quad cand \longleftarrow competitors(p_i, \mathbf{m}) \cup \{p_i\}$
12: $\quad\quad\quad\quad$ // ▷ *maximality check (Property 2.2)*
13: $\quad\quad\quad$ **if** $cand = competitors(cand, \{p_1, \ldots, p_i\})$ **then**
14: $\quad\quad\quad\quad candidates.insert(cand)$
15: $\quad\quad$ // ▷ *implicit case 3*
16: $\quad$ remove duplicates from $candidates$
17: $\quad \mathcal{M} \longleftarrow \mathcal{M} \cup candidates$
// ▷ p_i *is not competitive with any of* $\{p_1, \ldots, p_{(i-1)}\}$
18: $\quad$ **if** p_i not included in any clique so far **then**
19: $\quad\quad \mathcal{M}.insert(\{p_i\})$

Example. Hotel p_1 is processed first, since $\mathcal{M}$ is empty, singleton CSC $\{p_1\}$ is added to $\mathcal{M}$ in lines 18-19. Hotel p_2 is not compeitive with any CSC in $\mathcal{M}$, singleton CSC $\{p_2\}$ is inserted in lines 18-19. Hotel p_3 is competitive with CSC 2 and not partially competitive with CSC 1. CSC 2 is updated to include hotel p_3 ($\mathcal{M} = \{1, 23\}$). Hotel p_4 is not competitive with CSC 1 and is partially competitive with CSC 23. In line 11, a new candidate CSC, 34 ($competitors(p_4, 23) \cup \{p_4\} = 3 \cup 4$) is generated. CSC 34 is maximal within set 1234 (verified in lines 13-14) and is hence included in $candidates$. There are no duplicate sets in $candidates$ ($\mathcal{M} = \{1, 23, 34\}$). Hotel p_5 is partially competitive with CSC 23 and competitive with CSC 34. CSC 34 is extended to include hotel p_5. When processing set 23, a new candidate CSC 35 is generated ($competitors(p_5, 23) \cup \{p_5\} = 3 \cup 5 = 35$). CSC 35 fails the maximality

Table 4.1: Run of OMCE Algorithm on hotels dataset

p_i	$candidates$	$\mathcal{M}$
p_1	$\emptyset$	$\{ 1 \}$
p_2	$\emptyset$	$\{1, 2 \}$
p_3	$\emptyset$	$\{1, 23 \}$
p_4	$\{34\}$	$\{1, 23, 34 \}$
p_5	~~$\{35\}$~~	$\{1, 23, 345 \}$
p_6	$\emptyset$	$\{1, 23, 345, 6 \}$
p_7	$\emptyset$	$\{1, 23, 345, 67 \}$
p_8	$\emptyset$	$\{1, 23, 345, 678 \}$

check in line 13 since CSC 35 can be extended by including hotel p_4 and is hence not included in the candidate set. The run of the OMCE algorithm on the hotels dataset is detailed in Table 4.1.

Theorem 4.1

Given a threshold ε and the skyline set $\mathcal{S}$, the OMCE algorithm (Algorithm 1) enumerates all MCSCs ($\mathcal{M}$) of $\mathcal{S}$ in time $\mathcal{O}(d|\mathcal{S}|^2|\mathcal{M}|)$

Proof. To prove the correctness of the OMCE algorithm, we prove that all MCSCs of a set of skyline points S can be generated from the set of MCSCs of set of skyline points $S \setminus \{p\}$ where p is a skyline point in S using the OMCE algorithm, specifically following the three cases identified in Section 4.1 (illustrated in Fig. 4.2). Let $\mathbf{m}$ be a MCSC of set S and let p be a skyline point in S. There are two scenarios

- $p \notin \mathbf{m}$: If p is not included in $\mathbf{m}$, then $\mathbf{m}$ only contains mutually competitive skyline points from $S \setminus \{p\}$. Furthermore, $\mathbf{m}$ is maximal within $S \setminus \{p\}$ since if it were not maximal, then it would not be maximal in set S. Therefore, the OMCE algorithm generates all MCSCs of S that do not include p by processing MCSCs of set $S \setminus \{p\}$ that fall under case 3 (Fig. 4.2).
- $p \in \mathbf{m}$: Set $\mathbf{m} \setminus \{p\}$ consists of a set of mutually competitive skyline points in $S \setminus \{p\}$. Therefore, set $\mathbf{m} \setminus \{p\}$ is either a MCSC of set $S \setminus \{p\}$ or it is a subset of some MCSC of set $S \setminus \{p\}$. Cases 1 and 2 (Fig. 4.2) are capable of generating all MCSCs of set S that contain p from all MCSCs of set $S \setminus \{p\}$.

This concludes the proof of correctness. In the outer loop, the algorithm iterates over all points in the skyline set ($|\mathcal{S}|$) and the inner loop iterates over all MCSCs $|\mathcal{M}|$. Within each iteration, the algorithm performs competitiveness and maximality checks ($d.|\mathcal{S}|$). Duplicate elimination in line 16 can be performed by using a hash table to store $candidates$[1]. Therefore, the time complexity of Algorithm 1 is $\mathcal{O}(d|\mathcal{S}|^2|\mathcal{MC}|)$. □

[1]Hashing can also be performed efficiently by storing just the lower bounds of the mbbs of the MCSCs (Lemma 6.1)

Chapter 5

Optimisations and Pruning

The algorithm in [Tsukiyama et al., 1977] eliminated duplicate and non-maximal cliques by enforcing a partial order over maximal cliques. In the next section, we detail how we instead use spatial information of our data and provide optimisationsthat (1) address the efficiency of the competitiveness metric and maximality check (line 13) and (2) that prune the number of CSCs that we have to process given a skyline point p_i in line 5 of Algorithm 1.

5.1 Partitioning

It may be possible to partition the skyline set into a family of subsets $P = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ such that no point in one subset ($\mathbf{s}_i$) is competitive with any other point in the other subsets. This can be achieved using the following theorem (Theorem 5.1).

Theorem 5.1

If $\mathcal{S} = \{p_1, \ldots, p_m\}$ is the set of skyline points sorted by some dimension (say a) i.e. $p_1[a] \leq p_2[a] \leq \ldots \leq p_m[a]$, then for all points p_i $(1 \leq i < m)$

$$|p_i[a] - p_{i+1}[a]| > \varepsilon \implies \forall_{\substack{i_1 \leq i \\ i_2 > i}} \neg comp(p_{i_1}, p_{i_2})$$

Proof. Let $\mathcal{S} = \{p_1, p_2, p_3, \ldots, p_m\}$ be the set of skyline points sorted by dimension a, i.e. $p_1[a] \leq p_2[a] \leq p_3[a] \leq \ldots \leq p_m[a]$. Let p_i $(1 \leq i < m)$ be a skyline point such that $|p_i[a] - p_{i+1}[a]| > \varepsilon$,

then

$$
\begin{aligned}
|p_i[a] - p_{i+1}[a]| > \varepsilon &\iff p_{i+1}[a] - p_i[a] > \varepsilon && \text{(sorted)} \\
&\iff \forall_{i_2 > i}\, p_{i_2}[a] - p_i[a] > \varepsilon \wedge \forall_{i_1 < i}\, p_{i+1}[a] - p_{i_1}[a] > \varepsilon \\
&\iff \forall_{i_1 \leq < i_2}\, p_{i_2}[a] - p_{i_1}[a] > \varepsilon \\
&\iff \forall_{i_1 \leq i < i_2}\, \neg comp(p_{i_1}, p_{i_2})
\end{aligned}
$$

□

Theorem 5.1 states that if the skyline points are sorted on some dimension (say a) and there exist two consecutive points in the sorted set such that the difference between those two points in dimension a is greater than ε, then the set can be partitioned into two subsets (split between the two consecutive points) such that no point in the first subset is competitive with any other point in the second subset. If $\mathcal{C}(\mathbf{s})$ is the set of all CSCs of the points in the subset $\mathbf{s}$ and the skyline set is partitioned into a family of subsets $P = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ using Theorem 5.1 then,

$$\mathcal{C}(\mathcal{S}) = \bigcup_{\mathbf{s}_i \in P} \mathcal{C}(\mathbf{s}_i)$$

A similar statement can be made for MCSCs. Effectively, each subset of the partition can be treated as an independent problem. To partition the dataset, we first sort the skyline set by the first dimension and then partition the skyline set into a family of subsets using Theorem 5.1. Then, we repeat the partitioning on each of the subsets sorting by the second dimension and so on.

Example. The skyline set of the hotels dataset is sorted by the first dimension (Price) yielding order: 12345678. Theorem 5.1 is not applicable within this set since no two consecutive points differ by more than ε accross Price. The resulting partition after processing the first dimension is $\{\mathcal{S}\}$. Next, each subset of the partition is sorted by the second dimension (Distance). There is only one subset in the partition (i.e. $\mathcal{S}$); this subset is sorted by attribute Distance yielding the order: 87654321. Consecutive points (p_6, p_5) and (p_2, p_1) differ by more than ε across the second dimension. This set is then partitioned into subsets: $\{876, 5432, 1\}$. Each of these three subsets can be treated as independent datasets when applying the OMCE algorithm.

The cost of sorting an n-sized set of points using one dimension is $O(n \log n)$. Therefore, the time complexity for partitioning is $O(d|\mathcal{S}| \log|\mathcal{S}|)$. Note that in the description of the Partitioning algorithm, we stop when one iteration through each dimension has finished, when in fact, it is also possible to keep partitioning the skyline set until the partition sets have converged. For instance, assume a skyline set has been partitioned using the first dimension; when the subsequent partition sets are partitioned using the other dimensions, points are eliminated from the partition set. Therefore, consecutive points (when sorted by the first dimension) whose difference across the first dimension was less than ε in the partition

Algorithm 2 Partitioning Algorithm

Input: $\mathcal{S} = \{p_1, p_2, \ldots, p_m\}, \varepsilon$
$P \longleftarrow$ PARTITION$(\{\mathcal{S}\}, 1)$

function PARTITION$(P = \{\mathbf{s}_1, \ldots, \mathbf{s}_p\}, a)$
    if $a > d$ **then return** P
    $new\_P \longleftarrow \{\}$
    for all $\mathbf{s} \in P$ **do**
        sort(s, a) // ▷ *sort by dimension* a
        $new\_P \longleftarrow new\_P \cup$ THEOREM 5.1$(\mathbf{s}, a)$
    return PARTITION$(new_P, a + 1)$

// ▷ *Iteratively partitions* $\mathbf{s}$ *using Theorem 5.1*
function THEOREM 5.1$(\mathbf{s} = \{p_{i_1}, \ldots, p_{i_k}\}, d)$
    if s is empty **then return** $\{\}$
    $j \longleftarrow \min_j |p_{i_j}[d] - p_{i_{(j+1)}}[d]| > \varepsilon$ if exists else i_k
    return $\{\mathbf{s}[:j]\} \cup$ THEOREM 5.1$(\mathbf{s}[j+1:], d)$

set may not appear in the same partition set after other dimensions have been processed. The new pairs of consecutive points may differ by more than ε across the first dimension. In the theoretical worst case, to achieve convergence (no change in partition sets upon further application of the partitioning algorithm), the partition algorithm may need to be applied $|\mathcal{S}|$ times. Therefore, the time complexity of the partitioning algorithm till convergence is $O(d|\mathcal{S}|^2 \log|\mathcal{S}|)$. However, in practice, we found that the partition sets converge in a couple of iterations (Subsection 7.4.4).

Partitioning can decrease space requirements by the OMCE algorithm since only the fraction of the skyline set and MCSCs that correspond to the partition set being processed need to be stored in the main memory. Furthermore, each partition can be processed in parallel.

5.2 Minimum Bounding Boxes

The efficiency of the algorithms proposed in the previous section rely on the implementation of the $comp(,)$ and $parcomp(,)$ operators. To improve efficiency, we utilize the well-known concept of axis-parallel minimum bounding boxes (mbb).

Definition 5.1: Minimum Bounding Box

The **minimum bounding box** of a set of d-dimensional points $\mathbf{s}$ ($mbb(\mathbf{s})$) is the smallest d-dimensional axis-parallel box that contains all points in set $\mathbf{s}$. This bounding box is characterized by two d-dimensional points - lower (lb) and upper (ub) bounds. These bounds are given by the

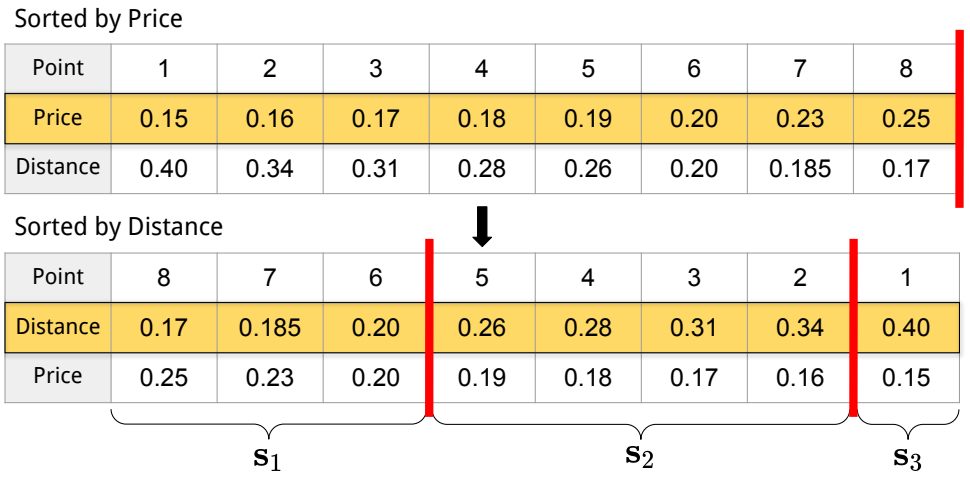


Figure 5.1: Partitioning hotels dataset using Theorem 5.1

following equations:

$$\forall_{a\in\mathcal{A}} \qquad mbb(\mathbf{s}).lb[a] = \min_{p\in\mathbf{s}} p[a]$$
$$mbb(\mathbf{s}).ub[a] = \max_{p\in\mathbf{s}} p[a]$$

Additionally, we define the size of a bounding box ($mbb.size$) to be the Chebyshev distance between the upper and lower bounds of the box. The minimum bounding box of a set $\mathbf{s}$ and the Chebyshev distances between points in $\mathbf{s}$ have an interesting relation (Property 5.1): the size of the bounding box of set $\mathbf{s}$ is the Chebyshev distance between the farthest pair of possible points in the box.

Property 5.1. Let $\mathbf{s}$ be a set of d-dimensional points, then

$$\max_{p_i,p_j\in\mathbf{s}} ||p_i - p_j||_\infty = mbb(\mathbf{s}).size$$

Proof. Let $\mathbf{s}$ be a CSC and x be the size of the mbb of $\mathbf{s}$ ($mbb(\mathbf{s}).size = x$), then

$$\begin{aligned} mbb(\mathbf{s}).size &= \max_{a\in\mathcal{A}} mbb(\mathbf{s}).ub - mbb(\mathbf{s}).lb \\ &= \max_{a\in\mathcal{A}} \max_{p_i,p_j\in\mathbf{s}} |p_i[a] - p_j[a]| \\ &= \max_{p_i,p_j\in\mathbf{s}} \max_{a\in\mathcal{A}} |p_i[a] - p_j[a]| \\ &= \max_{p_i,p_j\in\mathbf{s}} ||p_i - p_j||_\infty \end{aligned}$$

□

An important corollary of Property 5.1 is: A subset of the skyline set $\mathbf{s}$ is competitive if and only if the size of its mbb is at most ε. The bounding box can be used to determine if the set is competitive (Corollary 5.1).

Corollary 5.1. A subset of the skyline set $\mathbf{c}$ is competitive if and only if

$$mbb(\mathbf{c}).size \leq \varepsilon$$

Proof. The proof follows from Property 5.1. Let $\mathbf{c}$ be a CSC. Then,

$$mbb(\mathbf{c}).size = \max_{p_i,p_j \in \mathbf{c}} ||p_i - p_j||_\infty \leq \varepsilon$$

□

Property 5.1 and Corollary 5.1 lead to the following corollaries that can be used to efficiently implement the $comp(p, \mathbf{c})$ and $parcomp(p, \mathbf{c})$ operators

Corollary 5.2. A skyline point p is competitive with CSC $\mathbf{c}$ if and only if the size of the mbb of set $\mathbf{c} \cup \{p\}$ is at most ε

$$comp(p, \mathbf{c}) \iff mbb(\mathbf{c} \cup \{p\}).size \leq \varepsilon$$

Proof. The proof follows from Property 5.1. Let $\mathbf{c}$ be a CSC. Then,

$$mbb(\mathbf{c}).size = \max_{p_i,p_j \in \mathbf{c}} ||p_i - p_j||_\infty \leq \varepsilon$$

□

Corollary 5.3. If a skyline point p is partially competitive with CSC $\mathbf{c}$ then the size of the mbb of set $\mathbf{c} \cup \{p\}$ is at most 2ε

$$parcomp(p, \mathbf{cc}) \implies mbb(\mathbf{c} \cup \{p\}).size \leq 2\varepsilon$$

Proof. The proof of this property relies on proving the following statement: Let p, q, r be three skyline points such that $comp(p, q)$ and $comp(q, r)$, then $||p - r||_\infty \leq 2\varepsilon$ (we call this statement the **transitive bound property.**)

$$comp(p, q) \iff \forall_{a \in \mathcal{A}} |p[a] - q[a]| \leq \varepsilon$$
$$comp(q, r) \iff \forall_{a \in \mathcal{A}} |q[a] - r[a]| \leq \varepsilon$$

The projections of points p, q, r on dimension a are collinear and can have at most six configurations. These six possible permutations of the projects of points p, q, r on dimension a are shown in Fig. 5.2.

In four of the six cases ((b)–(d)), points p and r differ by at most ε, and in the remaining two cases ((a), (f)) they differ by at most 2ε. Therefore, for any given dimension a, when $comp(p, q)$ and $comp(q, r)$, points p and r differ by at most 2ε. We have hence ascertained that $||p - r||_\infty \leq 2\varepsilon$. The rest of the proof follows from Corollary 5.1.

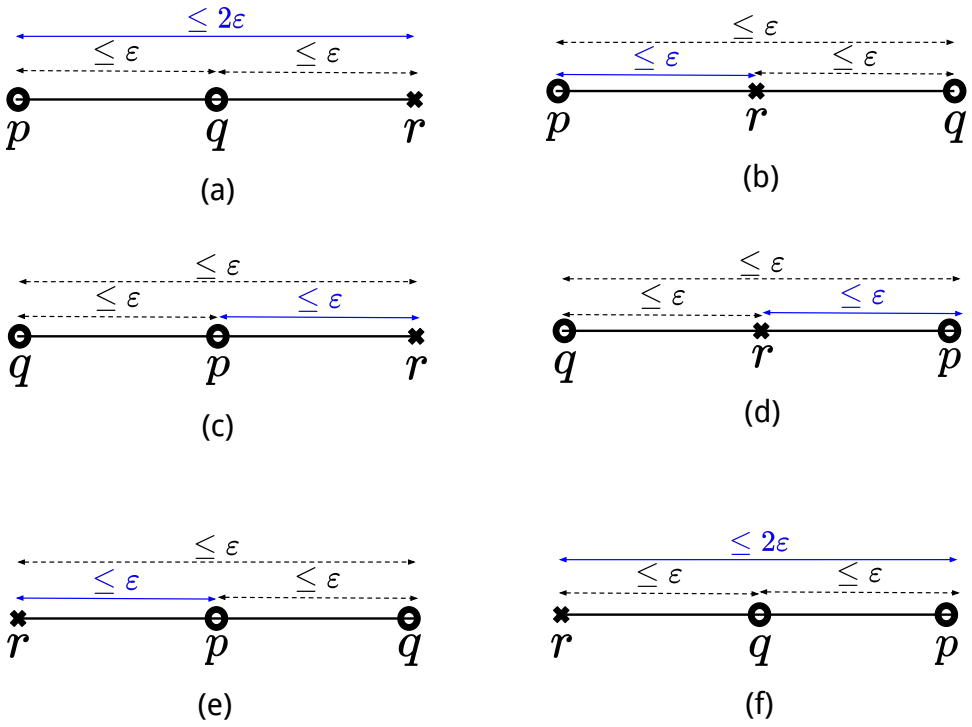


Figure 5.2: Projections of skyline points p, q, r on dimension a where $comp(p, q)$ and $comp(q, r)$. In cases (b)–(d), $p[a]$ and $r[a]$ differ by at mos ε and in cases (a) and (f), $p[a]$ and $r[a]$ differ by at most 2ε.

$$
\begin{aligned}
parcomp(p, \mathbf{c}) &\iff \exists_{q \in \mathbf{c}}\ comp(p, q) && \text{(by definition)} \\
&\iff \forall_{r \in \mathbf{c}}\ \exists_{q \in \mathbf{c}}\ comp(p, q) \wedge comp(q, r) \\
&\implies \forall_{r \in \mathbf{c}}\ ||p - r||_\infty \leq 2\varepsilon && \text{(transitive bound property)} \\
&\implies \forall_{x,y \in \mathbf{c} \cup \{p\}}\ ||x - y||_\infty \leq 2\varepsilon && (\mathbf{c} \text{ is competitive}) \\
&\implies mbb(\mathbf{c} \cup \{p\}).size \leq 2\varepsilon && \text{(Corollary 5.1)}
\end{aligned}
$$

□

Notice the single direction of the implication in Corollary 5.3. The proofs for Corollaries 5.2 and 5.3 follow from Corollary 5.1. The corollaries (Corollaries 5.2 and 5.3) are used to implement the case logic, specifically case 1 and case 2 in the algorithm. Let p be a skyline point and $\mathbf{c}$ be a CSC, then:

case 1 $comp(p, \mathbf{c})$: To check if p is competitive with $\mathbf{c}$, it is sufficient to check if $mbb(\mathbf{c} \cup \{p\}).size \leq \varepsilon$ (Corollary 5.2).

case 2 $parcomp(p, \mathbf{c})$: If $mbb(\mathbf{c} \cup \{p\}).size > 2\varepsilon$, then point p is not competitive with any point in CSC $\mathbf{c}$ (Corollary 5.3). However, if $mbb(\mathbf{c} \cup \{p\}).size \leq 2\varepsilon$, then p may be competitive with some point in $\mathbf{c}$, a brute force check is necessary to ascertain partial competitiveness. Corollary 5.3 helps prune case 2.

Using mbbs reduces the number of distance computations performed. For instance, to check if a point p is competitive with a CSC $\mathbf{c}$ of cardinality n, the brute force method is to perform n distance computations. The time complexity is $\mathcal{O}(nd)$. When using mbbs, we only need one distance computation leading to $\mathcal{O}(d)$ time complexity. Using mbbs reduces the complexity of the OMCE algorithm by a factor of $|\mathcal{S}|$. Furthermore, when we partition the dataset, we can establish early on if the parition set is competitive — If the size of the mbb of the partition set is at most ε, then the partition set is a MCSC. Then there is no need to run the OMCE algorithm on the partition set.

5.3 Indexing

Bounding boxes also provide other advantages. For instance, we use $crange(p)$ to denote the competitive range of point p. $crange(p)$ is a d-dimensional box that contains all the points competitive with p. $crange(p)$ is given by the following equation:

$$crange(p).lb[a] = \max(0, p[a] - \varepsilon)$$
$$crange(p).ub[a] = \min(1, p[a] + \varepsilon)$$

In other words $competitors(p, \mathcal{S}) = \{p' | p' \in \mathcal{S} \; and \; p' \in crange(p)\}$. Similarly, the notation can be extended to work for CSCs. Let $\mathbf{c}$ be a CSC, then $crange(\mathbf{c})$ is given by:

$$crange(\mathbf{c}).lb[a] = \max(0, mbb(\mathbf{c}).ub[a] - \varepsilon)$$
$$crange(\mathbf{c}).ub[a] = \min(1, mbb(\mathbf{c}).lb[a] + \varepsilon)$$

We use $pcrange(p)$ to denote the partial competitive range of p. $pcrange(p)$ is given by the following equation:

$$pcrange(p).lb[a] = \max(0, p[a] - 2\varepsilon)$$
$$pcrange(p).ub[a] = \min(1, p[a] + 2\varepsilon)$$

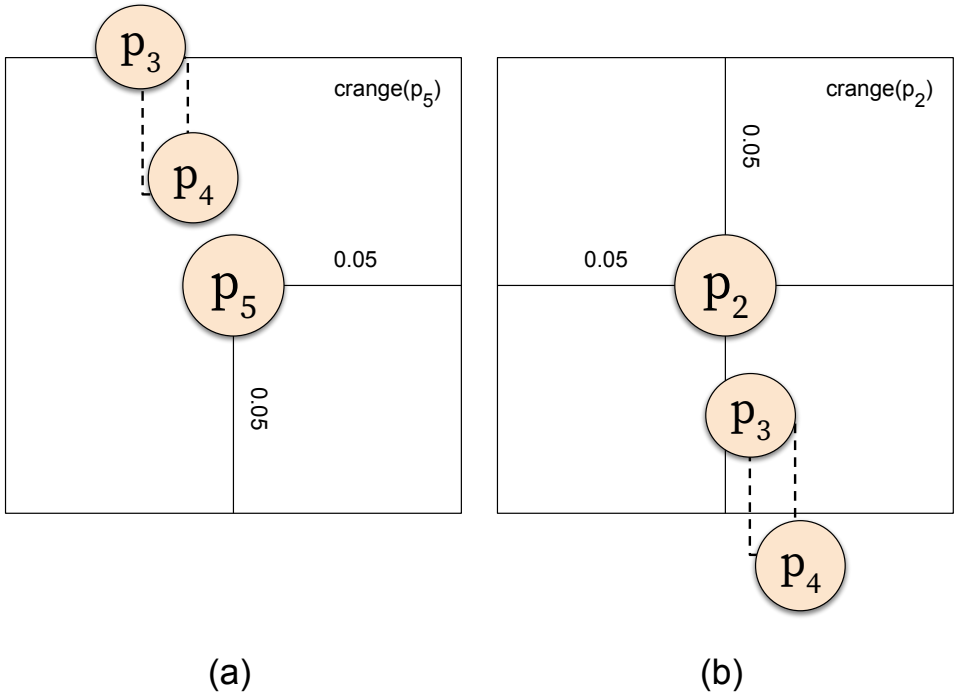


Figure 5.3: Example demonstrates utility of $crange()$ boxes. (a) Hotel p_5 if competitive with both hotels in set 34. Hotel 34 lies within $crange(p_5)$. (b) Hotels p_2 is competitive with hotel p_3 but not with hotel p_4, therefore set 34 intersects $crange(p_2)$ and lies within $pcrange(p_2)$ ($pcrange(p_2)$ not shown)

If a CSC **c** is partially competitive with p, then it is contained in $pcrange(p)$. But a stricter condition for partial competitiveness is: If a CSC **c** is partially competitive with skyline point p, then it **intersects** $crange(p)$. Given the spatial nature of our data and the introduction of mbbs, it is possible to use a spatial index to store both the skyline set and the MCSCs. In this thesis, we assume that the MCSCs are stored in an R-Tree, indexed by their mbbs.

In the OMCE algorithm (Algorithm 1), we are only interested in processing cliques that are partially competitive with point p. Instead of iterating through all the MCSCs, we can use a range query on the R-Tree that stores the MCSCs to retrieve only those sets that *intersect* box $crange(p)$ (strict) or are contained in box $pcrange(p)$ (lose). All sets that fall within $crange(p)$ are extended (case 1), and all other sets are further processed in lines 10-14. Additionally, in line 13, a maximality check needs to be performed. We optimize this operation by using a range query on the R-Tree storing skyline points using range box $crange(cand)$. Furthermore, MCSCs can be uniquely determined by their mbbs (Lemma 6.1, Chapter 6); therefore, to eliminate duplicates in the $candidates$ set, we can hash the mbbs of the MCSCs instead of the points in the cliques.

Chapter 6

Complexity Analysis

In this section, we attempt to bound the number of CSCs and MCSCs ($|\mathcal{C}|$ and $|\mathcal{M}|$ respectively). The worst case cardinality of $\mathcal{C}$ is $\mathcal{O}(2^{|\mathcal{S}|})$. This worst case occurs when every skyline point is competitive with every other skyline point. The worst case cardinality of $\mathcal{M}$ is harder to bound. [Moon and Moser, 1965] showed that for any n-vertex graph, the number of maximal cliques is at most $\mathcal{O}(3^{n/3})$, exponential in n. However, we show that the worst-case number of MCSCs is $\mathcal{O}(|\mathcal{S}|^d)$, polynomial in the number of skyline points.

Theorem 6.1

The worst-case number of MCSCs ($|\mathcal{M}|$) is $\mathcal{O}(|\mathcal{S}|^d)$.

Proof. To derive this bound, we first prove the following lemma.

Lemma 6.1

Let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two distinct MCSCs. Then, the lower (and upper) bounds of the mbbs containing the two MCSCs are not equal.

$$\forall_{\substack{\mathbf{m}_1,\mathbf{m}_2\in\mathcal{M}\\ \mathbf{m}_1\neq\mathbf{m}_2}} \quad mbb(\mathbf{m}_1).lb \neq mbb(\mathbf{m}_2).lb \quad mbb(\mathbf{m}_1).ub \neq mbb(\mathbf{m}_2).ub$$

Proof. We prove this lemma by assuming the opposite is true. Let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two distinct maximal competitive cliques and lb be the d-dimensional point that is the lower bound of the minimum bounding boxes of the two sets.

$$lb = mbb(\mathbf{m}_1).lb = mbb(\mathbf{m}_2).lb$$

Let bb be the ε-sized box with lb as the lower bound such that the width of the box across every dimension is ε i.e.

$$\begin{aligned} \forall_{a \in \mathcal{A}} \qquad bb.lb[a] &= lb[a] \\ bb.ub[a] &= lb[a] + \varepsilon \end{aligned}$$

Then, according to Property 5.1 and Corollary 5.1,

$$\begin{aligned} mbb(\mathbf{m}_1) \in bb \wedge mbb(\mathbf{m}_2) \in bb & \\ \iff & (mbb(\mathbf{m}_1) \cup mbb(\mathbf{m}_2)) \in bb \\ \iff & (\mathbf{m}_1 \cup \mathbf{m}_2) \in bb \\ \iff & mbb(\mathbf{m}_1 \cup \mathbf{m}_2).\texttt{size} \leq \varepsilon \\ \iff & \mathbf{m}_1 \cup \mathbf{m}_2 \text{ is a CSC} \end{aligned}$$

This is a contradiction. If $\mathbf{m}_1 \cup \mathbf{m}_2$ is competitive and $\mathbf{m}_1$ and $\mathbf{m}_2$ are distinct, then neither $\mathbf{m}_1$ nor $\mathbf{m}_2$ can be maximal. □

From Lemma 6.1, it follows that the number of MCSCs is therefore upper bounded by the number of possible lower bounds of mbbs containing subsets of the skyline set. The number of possible lower bounds is upper bounded by $|\mathcal{S}|^d$ since the lower bound is a d-dimensional point, and any of the skyline points can contribute to the lower bound in each dimension. This concludes the proof of Theorem 6.1. □

The result of Theorem 6.1 can also independently be derived as a result of a property exhibited by competitiveness graphs: competitiveness graphs have a bounded *boxicity* of d where d is the dimensionality of the dataset. The boxicity [Roberts, 1969] of a graph is the minimum dimensionality of the space in which each vertex can be represented as an axis-parallel rectangle/box, and an edge exists between two vertices iff their corresponding boxes intersect. In our case, the vertices are skyline points, and the box corresponding to a skyline point is an ε-sized axis-parallel d-dimensional cube centred at the point. The boxed representation of the hotels dataset is shown in Fig. 6.1. [Spinrad, 2003] showed that graphs with a bounded boxicity of b (boxicity is at most b) have at most $\mathcal{O}((2n)^b)$ maximal cliques where n is the number of vertices.

Note that the proof of Theorem 6.1 and Lemma 6.1 do not take into account the fact that all the points in question are skyline points. We show that the bound derived in Theorem 6.1 is reasonably tight for skyline points by constructing a skyline set such that the number of MCSCs is $\Omega\left((|\mathcal{S}|/2(d-1))^{d-1}\right)$

Theorem 6.2

The worst case number of MCSCs ($|\mathcal{M}|$) is $\Omega\left(\left(\frac{|\mathcal{S}|}{2(d-1)}\right)^{d-1}\right)$

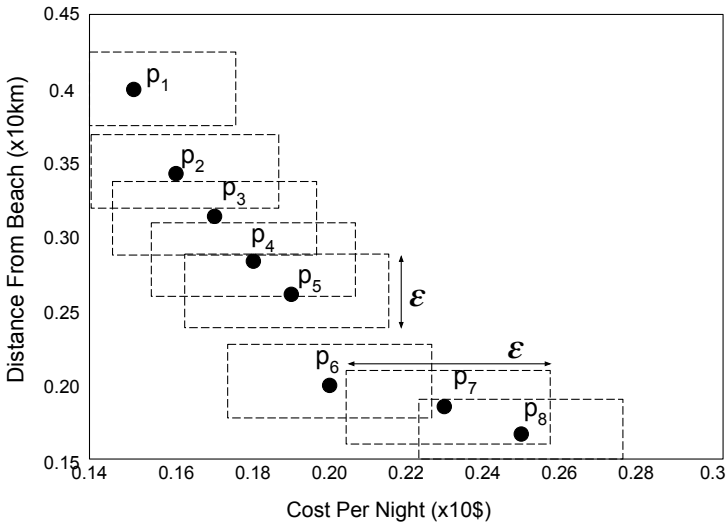


Figure 6.1: Boxed representation of hotels dataset. Note that the boxes are squares.

Proof. To prove Theorem 6.2, (1) we first construct a n-sized $(d-1)$-dimensional dataset such the number of MCSCs is $(n/2(d-1))^{d-1}$ when competitiveness is not limited to just skyline points and (2) then show that any general n-sized $(d-1)$-dimensional dataset can be converted to an n-sized d-dimensional dataset of only skyline points with the Chebyshev distances conserved.

Consider a set of n $(d-1)$-dimensional points (D) constructed using multiples of unit vectors along each of the $(d-1)$ dimensions. Along each dimension, there are $n/(d-1)$ points equally spaced between $[-\varepsilon, \varepsilon]$. For instance, when $n = 8$ and $d = 3$, we construct the following set of 8 2-dimensional points.

$$D = \left\{-\varepsilon e_1, -\frac{\varepsilon}{3}e_1, +\frac{\varepsilon}{3}e_1, +\varepsilon e_1, -\varepsilon e_2, -\frac{\varepsilon}{3}e_2, +\frac{\varepsilon}{3}e_2, +\varepsilon e_2\right\}$$

where,

$$e_1 = \begin{pmatrix}1\\0\end{pmatrix} \text{ and } e_2 = \begin{pmatrix}0\\1\end{pmatrix}$$

This dataset can be transformed into an n-sized d-dimensional dataset of only skyline points by adding another dimension such that the Chebyshev distances between every pair of points are conserved. To perform this transformation, we partition the dataset into layers of skyline points[Liu et al., 2015] where the first layer is the skyline set of D ($\mathcal{S}$), the second layer is the skyline of $(D - \mathcal{S})$ and so on. This provides a partial ordering of the points in D. Every point in the same skyline layer is provided with the same value of the dth dimension, and points in higher skyline layers are provided with smaller values to ensure that they are not dominated. Finally, all the values of the dth dimension range within the smallest Chebyshev distance between any two points in D. When $n = 8$ and $d = 3$, the transformed dataset

resembles

$$\left\{ \begin{pmatrix} -\varepsilon \\ 0 \\ \varepsilon/4 \end{pmatrix}, \begin{pmatrix} -\varepsilon/3 \\ 0 \\ \varepsilon/5 \end{pmatrix}, \begin{pmatrix} \varepsilon/3 \\ 0 \\ \varepsilon/6 \end{pmatrix}, \begin{pmatrix} \varepsilon \\ 0 \\ \varepsilon/7 \end{pmatrix}, \begin{pmatrix} 0 \\ -\varepsilon \\ \varepsilon/4 \end{pmatrix}, \begin{pmatrix} 0 \\ -\varepsilon/3 \\ \varepsilon/5 \end{pmatrix}, \begin{pmatrix} 0 \\ \varepsilon/3 \\ \varepsilon/6 \end{pmatrix}, \begin{pmatrix} 0 \\ \varepsilon \\ \varepsilon/7 \end{pmatrix} \right\}$$

The first skyline layer of D is 15, the second is 26, the third is 37, and the final layer is 48. The shortest Chebyshev distance is $2\varepsilon/3$; the 3rd dimension is provided values amongst $\varepsilon/4, \varepsilon/5, \varepsilon/6$ or $\varepsilon/7$ based on the layer of the point. Note that the maximum Chebyshev distance across the 3rd dimension is $3\varepsilon/28$, which conserves the Chebyshev distances between all the points in D. This dataset satisfies the following properties:

1. Every point along one of the first $d-1$ dimensions is competitive with all points along all other dimensions, i.e. the distance between any two points along different dimensions is at most ε.

2. Within the set of points along a particular dimension (among the first $d-1$ dimensions), there are $n/2(d-1)$ MCSCs.

Using points (1) and (2), there are a total of $(n/2(d-1))^{d-1}$ MCSCs each of size $n/2$ in this constructed dataset. This concludes the proof of Theorem 6.2. □

Chapter 7

Experimental Analysis

Because of the extent of redundancy amongst CSCs, we restrict our attention to MCSCs. We demonstrate the utility of MCSCs using three real datasets — UCars (Section 7.1), FIFA22 (Section 7.2) and Pokémon (Section 7.3). Then, we empirically analyse the effects of the dataset dimensionality (d), cardinality (n) and threshold (ε) on three aspects of MCSCs: (1) the number of MCSCs ($|\mathcal{M}|$), (2) the cardinality or size of MCSCs and (3) the overlap or visibility coefficient of MCSCs[1]. We measure the overlap of a MCSC ($\mathbf{m}$) by adapting the definition of visibility from [Wang et al., 2013]:

$$overlap(\mathbf{m}, \mathcal{M}) = \max_{\mathbf{m}' \in \mathcal{M}} \frac{|\mathbf{m} \cap \mathbf{m}'|}{|\mathbf{m}|}$$

The overlap of a MCSC is a measure of the redundancy of a MCSC. If a MCSC has high overlap, it indicates that another MCSC exists that contains most points within the MCSC.

[1]The codebase is public and open source [Source Code].

7.1 Case Study: Used Cars

The vehicles dataset is a set of 8K used cars scraped from the used cars website https://www.cardekho.com [Birla, 2021]. For each car, the dataset records 11 different attributes – make and model (Name), year of purchase (Year), Selling Price ($), number of kms driven (Driven (km)), Fuel (diesel/petrol), Transmission (manual/automatic), number of previous owners (# Owners), Mileage (kmpl), Engine (CC), Max Power (bhp) and the number of seats (Seats).

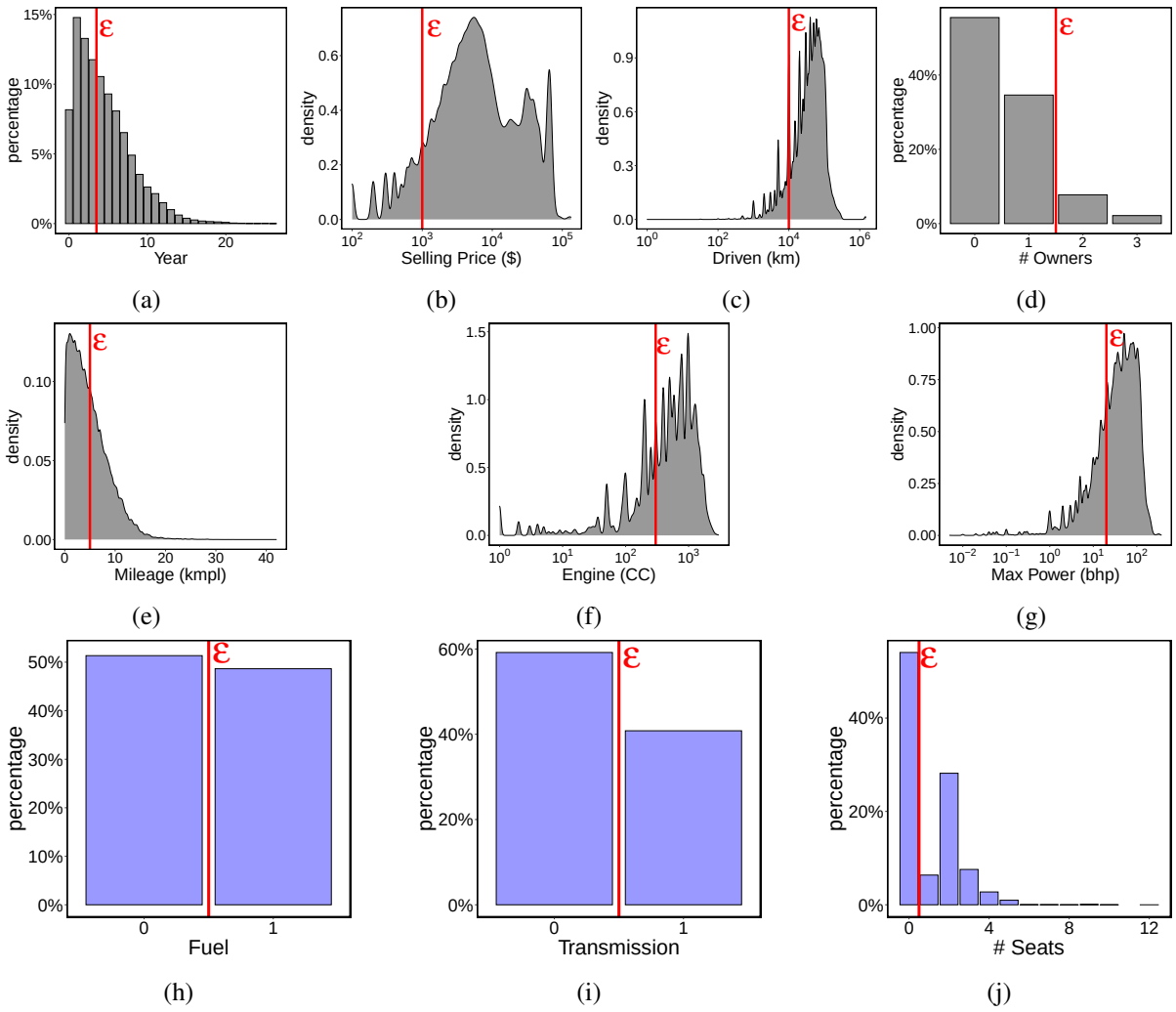


Figure 7.1: Distribution of Chebyshev distances between pairs of skyline points in the UCars dataset accorss each attribute. The red line indicates the threshold ε used for each attribute

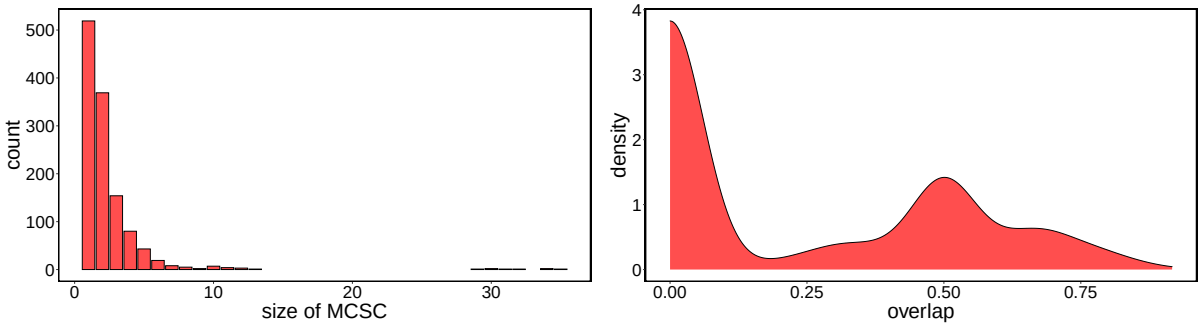
Not all attributes are relevant when computing the skyline. For instance, the better of petrol and diesel is subjective. However, these attributes can be incorporated when computing MCSCs. For instance, in this experiment, the Chebyshev distance between Petrol and Diesel is 1, i.e. petrol and diesel cars are incomparable. The distribution of pairwise Chebyshev distances between skyline points across each

attribute is shown in Fig. 7.1. Attributes like Transmission, Fuel and Seats partition the dataset. As discussed in Chapter 2, a different ε is specified for each attribute. Picking thresholds can be challenging; ε should be set to a value that a consumer might find insignificant. For instance, in the case of cars, a consumer might find a 20 bhp difference in power insignificant. Overall, the dataset has 2K skyline points and 1.2K MCSCs. A sample of MCSCs of the UCars dataset is shown in Fig. 7.3.

1. Notice that no two cars in the same MCSC have different transmissions, fuel types and the number of seats. These three attributes partition the dataset. For instance, set 14 consists of exactly one 14-seater car and set 2 consists of automatic, diesel-based 5-seater cars.

2. Additionally, the cliques have grouped cars in a logically pleasing way. For instance, clique 1 consists of the only vintage skyline car. Clique 2 consists of higher-end expensive, and powerful sport cars. Clique 3 consists of cars in the cheaper range that were manufactured around ten years ago with pretty good mileage. Clique 4 consists of popular classic oldies – old, cheap petrol-based cars with good mileage and little power. And so on.

Fig. 7.2a and Fig. 7.2b plot the distribution of cardinalities and overlaps of MCSCs respectively. Notice that a majority of the MCSCs are small and have almost zero overlap.



(a) Distribution of MCSC cardinalities or sizes

(b) Distribution of overlaps of MCSCs

Figure 7.2: Properties of MCSCs of the UCars dataset

MCSC ID	Name	Year	Selling Price ($)	Driven (km)	Fuel	Transmission	# Owners	Mileage (kmpl)	Engine (CC)	Max Power (bhp)	Seats
1	Ambassador Classic 2000 DSZ AC PS	1994	1300	100000	Diesel	Manual	2	12.8	1995	52	5
2	Audi A4 35 TDI Premium Plus	2018	38600	15000	Diesel	Automatic	1	18.25	1968	187.74	5
	Audi Q3 2.0 TDI Quattro Premium Plus	2017	37600	22000	Diesel	Automatic	1	15.73	1968	174.33	5
	BMW 5 Series 520d Luxury Line	2016	38600	12000	Diesel	Automatic	1	18.12	1995	190	5
3	Chevrolet Sail Hatchback LT ABS	2013	2600	70000	Diesel	Manual	1	22.1	1248	76.9	5
	Fiat Grande Punto Active (Diesel)	2012	2200	60000	Diesel	Manual	2	20.3	1248	75	5
	Ford Classic 1.4 Duratorq CLXI	2013	2600	70000	Diesel	Manual	2	19.68	1399	67	5
	Ford Figo Diesel Celebration Edition	2013	3000	70000	Diesel	Manual	1	20	1399	68.1	5
	Ford Figo Diesel LXI	2012	3000	60000	Diesel	Manual	1	20	1399	68	5
	Ford Figo Diesel ZXI	2011	2200	70000	Diesel	Manual	2	20	1399	68	5
	Maruti Ritz VDi	2012	2900	70000	Diesel	Manual	2	23.2	1248	73.94	5
	Tata Indica Vista TDI LX	2013	3000	70000	Diesel	Manual	2	19.4	1405	70	5
	Tata Indigo CS LS (TDI) BS-III	2014	2900	65000	Diesel	Manual	1	19.09	1405	69.01	5
4	Maruti Alto LX	2006	800	100000	Petrol	Manual	2	19.7	796	46.3	5
	Maruti Wagon R LX BSIII	2008	1000	100000	Petrol	Manual	3	17.3	1061	64	5
5	Force One EX	2018	11300	120000	Diesel	Manual	2	17	2650	80.84	7
	Mahindra Scorpio S2 7 Seater	2017	11300	110000	Diesel	Manual	1	15.4	2523	75	7
6	Ford EcoSport 1.5 Petrol Titanium BSIV	2019	11800	10000	Petrol	Manual	2	17	1497	121.31	5
	Ford EcoSport 1.5 Petrol Titanium Plus BSIV	2018	12100	15000	Petrol	Manual	1	17	1497	121.36	5
	Honda City i VTEC V	2017	11300	11688	Petrol	Manual	1	17.8	1497	117.3	5
	Hyundai Verna VTVT 1.6 SX	2017	12000	20000	Petrol	Manual	1	17	1591	121.3	5
	Nissan Kicks XL BSIV	2019	11400	14317	Petrol	Manual	1	14.23	1498	104.55	5
7	Ford Figo 1.2P Ambiente MT	2011	3200	25000	Petrol	Manual	1	18.16	1196	86.8	5
	Ford Figo Petrol EXI	2011	3000	15000	Petrol	Manual	1	15.6	1196	70	5
	Hyundai i10 Era	2010	2800	20000	Petrol	Manual	1	19.81	1086	68.05	5
	Maruti Ritz VXi	2011	3000	18500	Petrol	Manual	1	18.5	1197	85.8	5
	Maruti Swift VXI	2011	3700	20000	Petrol	Manual	2	18.6	1197	85.8	5
8	Ford Figo 1.5D Trend MT	2017	6600	56000	Diesel	Manual	1	25.83	1498	99	5
	Honda Amaze S Option i-DTEC	2018	7400	50000	Diesel	Manual	1	25.8	1498	98.6	5
	Maruti Ciaz VDi Plus SHVS	2015	7300	50000	Diesel	Manual	2	28.09	1248	88.5	5
9	Ford Figo Diesel LXI	2012	1800	70000	Diesel	Manual	4	20	1399	68	5
	Ford Figo Diesel Titanium	2011	2000	80000	Diesel	Manual	3	20	1399	68	5
10	Ford Freestyle Titanium Petrol BSIV	2020	5300	5000	Petrol	Manual	2	19	1194	94.68	5
	Honda Brio 1.2 VX MT	2017	6000	6750	Petrol	Manual	1	18.5	1198	86.8	5
	Maruti Swift LXI	2020	5900	1000	Petrol	Manual	1	21.21	1197	81.8	5
11	Maruti Baleno Delta 1.3	2017	8000	20000	Diesel	Manual	1	27.39	1248	74	5
	Maruti Swift Dzire VDI	2017	8600	20000	Diesel	Manual	1	28.4	1248	74.02	5
	Maruti Swift VDI	2019	8400	15000	Diesel	Manual	2	28.4	1248	74	5
12	Ford Aspire Titanium BSIV	2019	7600	20000	Petrol	Manual	1	19.4	1194	94.93	5
	Ford Aspire Trend Plus	2018	8200	15000	Petrol	Manual	1	20.4	1194	94.93	5
	Maruti Dzire LXI	2020	7300	15000	Petrol	Manual	1	23.26	1197	88.5	5
13	Tata Indica V2 DLE BSII	2009	600	100000	Diesel	Manual	2	17.2	1396	53.5	5
	Tata Indica Vista TDI LS	2012	1500	95000	Diesel	Manual	2	19.4	1405	70	5
14	Tata Winger Deluxe - Flat Roof (Non-AC)	2010	3100	50000	Diesel	Manual	1	10.71	1948	90	14
15	Hyundai Xcent 1.2 CRDi E Plus	2018	6200	25000	Diesel	Manual	1	25.4	1186	73.97	5
	Maruti Swift DDiS LDI	2017	6600	35000	Diesel	Manual	1	28.4	1248	74	5
	Maruti Swift Dzire Tour LDI	2017	6900	28832	Diesel	Manual	1	23.4	1248	74	5
	Maruti Swift Dzire VDI	2015	6400	30000	Diesel	Manual	1	26.59	1248	74	5
	Toyota Etios Liva GD	2015	6000	25000	Diesel	Manual	1	23.59	1364	67.04	5

Figure 7.3: Sample MCSCs of the UCars dataset

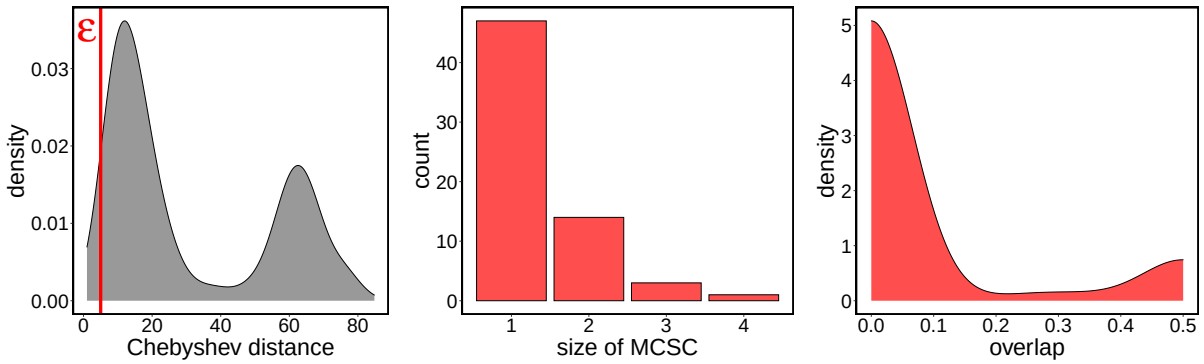
7.2 Case Study: FIFA Dataset

We study MCSCs of a dataset of football players scraped from https://sofifa.com [Leone, 2021]. The dataset consists of 19K football players rated out of 100 across 34 different attributes. The website partitions these attributes into seven categories — Attacking, Skill, Movement, Power, Mentality and Goalkeeping. The skyline set is computed by maximising across all seven attributes. There are 80 skyline football players. The Sofifa website statistics are commonly used when playing Fantasy Football. Building a team can be challenging when trying to fill a position. For this dataset, we set $\varepsilon = 5\%$. Furthermore, we use a custom competitiveness metric for the Position attribute that records the positions that each player plays. Two players are considered to be competitive if and only if they are competitive across all numerical attributes and play at least one common position. A sample of MCSCs of the FIFA22 dataset is shown in Fig. 7.4.

MCSC ID	Name	Positions	Attacking	Skill	Movement	Power	Mentality	Goalkeeping
1	A. Griezmann	ST, LW, RW	84	85.4	84.8	80.8	77.5	12.6
	H. Son	LM, CF, LW	80.2	80.8	86	77.8	72.83	10.6
2	Cristiano Ronaldo	ST, LW	87.6	83.6	85.4	87.2	74.33	11.6
3	E. Can	CM, CB, CDM	76.2	77	76.4	85.4	77.67	10.8
	L. Goretzka	CM, CDM	79.6	81.2	79.2	85.2	79.67	11.2
	M. Sabitzer	CM, CDM, CAM	76.2	79.2	79.4	81.6	76.83	13.2
	Paulinho	CM, CAM, CDM	78.4	76.4	76.6	85.6	79.67	11.4
4	F. Muslera	GK	18.2	24.6	66	51.2	35.67	78.2
	Jordi Masip	GK	21.2	24.2	69.4	47.8	31.16	73.8
5	M. Neuer	GK	24.8	33.8	57.4	56.8	43	88.8
6	Neymar Jr	LW, CAM	80.6	89.2	90.2	71.8	77	11.8
7	Sergio Ramos	CB	74.8	76.2	75	79.6	83.5	9.2

Figure 7.4: Sample MCSCs of the FIFA22 dataset

Fig. 7.5a plots the distribution of Chebyshev distances between pairs of skyline players. Notice that the number of competitive relationships is few. The dataset generates 65 MCSCs. Figs. 7.5b and 7.5c plots the distribution of MCSC cardinalities and overlaps. Most MCSCs are small and have low overlaps. In fact, most MCSCs are singletons and the few that are larger have players with indiscernible differences.



(a) Distribution of distances (b) MCSC cardinalities (c) overlaps of MCSCs

Figure 7.5: Properties of MCSCs and skyline set of the FIFA22 dataset. The red line in (a) depicts the value of ε used for FIFA22 dataset

7.3 Case Study: Pokémon Dataset

The Pokémon game franchise is based in a fictional universe where players are *Pokémon Trainers* — people who *collect* Pokémon, train them and battle each other using their Pokémon. Pokémon are unique creatures in this fictional universe that have supernatural abilities or *moves* based on the *type* (or types) of Pokémon they are. There are 18 different possible types, for instance *Water*, *Electric*, *Fire*, *Bug*. The moves that a Pokémon can have or learn are dependent on the type of the Pokémon. Fig. 7.6 shows the description of one Pokémon, Swampert. Swampert is both a water-type and a ground-type Pokémon. Therefore, all the moves that Swampert has are either water-type or ground-type moves.

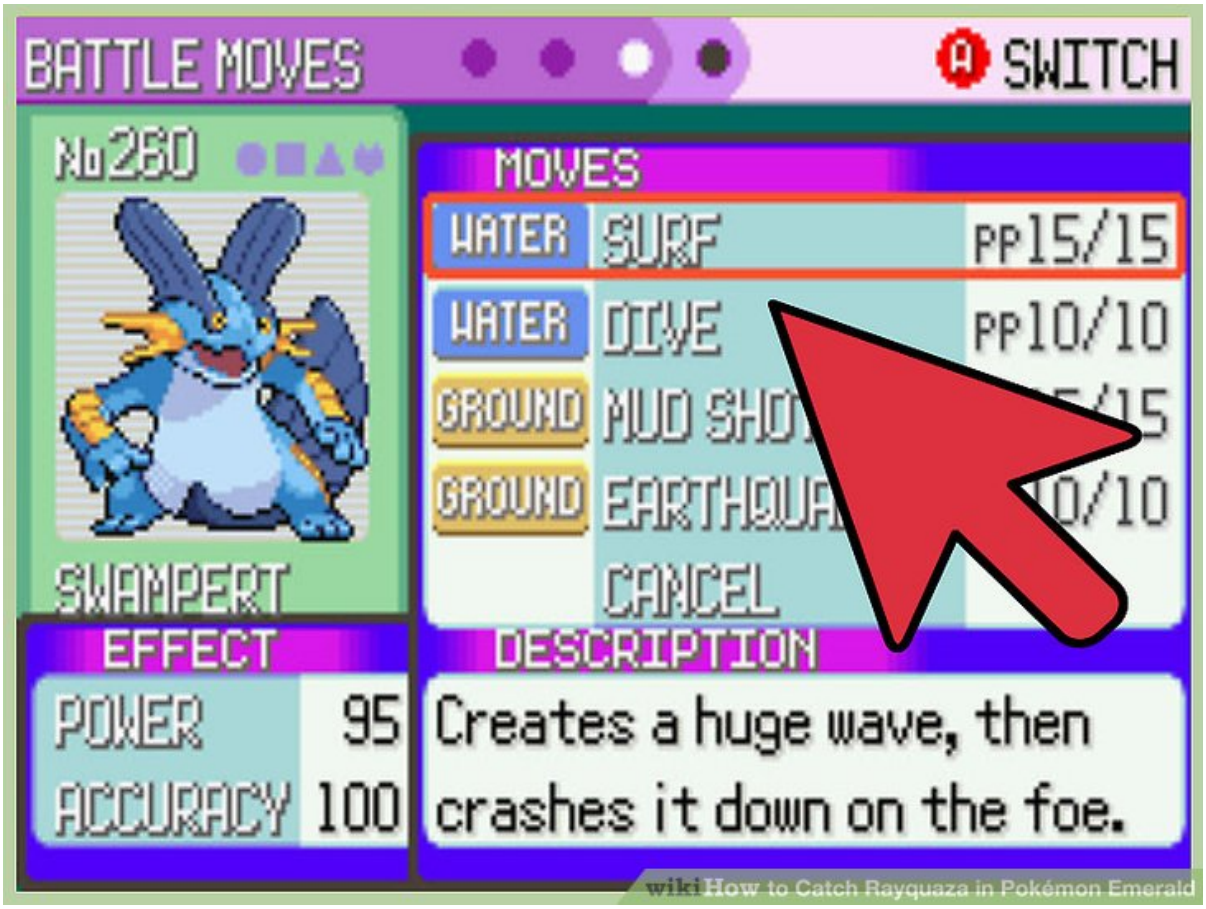


Figure 7.6: Pokémon move types. Swampert is a dual-type Pokémon (Ground + Water). Therefore, it is capable of learning most *Ground* and *Water* type moves. For instance, *SURF* is a water-type move, and *EARTHQUAKE* is a ground-type move. Source: www.wikihow.com

When a battle between two trainers begins, each trainer picks one Pokémon. Then, in turn, each trainer can either choose to switch out their Pokémon to a different one having seen the opponent's Pokémon or can pick a move that their chosen Pokémon has. The objective is to pick moves that can effectively inflict *damage*, i.e. reduce the *health points* (HP) of the opponent's Pokémon. The effectiveness of a Pokémon is dependent on its type — A water-type Pokémon is strong against Fire-

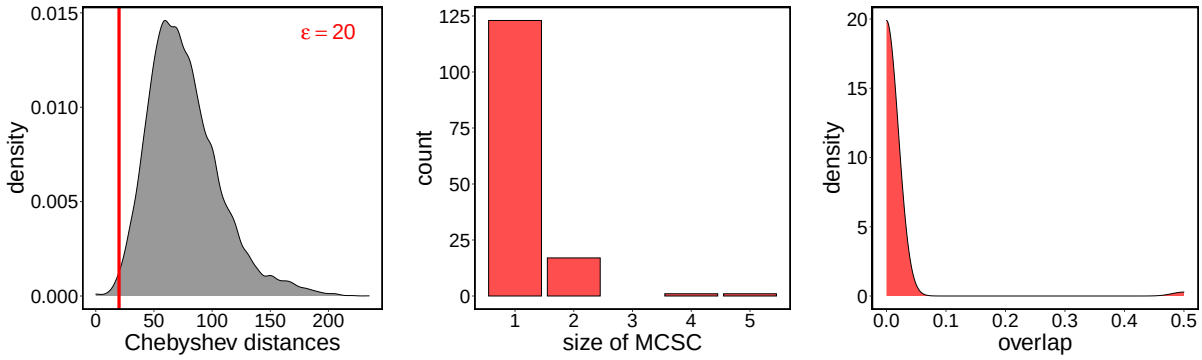
type Pokémon but weak against Grass-type and Electric-type Pokémon. The incentive to switch your Pokémon is to pick a Pokémon that is strong against the opponent's Pokémon. When a Pokémon gets knocked out (HP $= 0$), the trainer is forced to switch out their Pokémon to a different one. The game ends when one of the trainers has no Pokémon left. Therefore, the meta-objective of the game is to build a team of Pokémon of diverse types that are cumulatively effective against all types.

Figure 7.7: A Pokémon battle UI. Blaziken is a Fire-type Pokémon. The player has to pick amongst one of four moves

While the factors of the game are nuanced and the amount of *damage* that a Pokémon inflicts is dependent on a variety of factors[2], we use a dataset of all 800 Pokémon scraped from a variety of sources[Barradas, 2016]. For each Pokémon, the dataset records seven attributes — (1) the type(s) of the pokemon (Type 1, Type 2 if dual), (2) the maximum health points a Pokémon can have (HP), (3) the base attack power (Attack), (4) the base defence power (Defense), (5) the base special attack power (Sp. Atk), (6) the base special defence power (Sp. Def), and (7) the speed which determines which Pokémon attacks first (Speed). All the numerical attributes are given integer values less than 300. A Pokémon dominates another Pokémon if they share at least one type in common and the first Pokémon is stronger wrt all numerical attributes.

[2]To read about how *damage* is calculated, please see https://bulbapedia.bulbagarden.net/wiki/Damage

This dataset consists of 168 skyline Pokémon. Fig. 7.8a plots the distribution of Chebyshev distances between all pairs of skyline Pokémon. We compute all MCSCs where we set the threshold to ε. Similar to the FIFA22 dataset, we modify the competitiveness metric, two Pokémon are competitive only if they share at least one type in common. A sample of the MCSCs of the Pokémon dataset is shown in Fig. 7.9. A total of 142 MCSCs are generated. Most MCSCs are singletons (Fig. 7.8b), and all but one MCSC have an overlap of zero.



(a) Distribution of distances (b) MCSC cardinalities (c) overlap of MCSCs

Figure 7.8: Properties of MCSCs and skyline set of the Pokémon dataset. The red line in (a) depicts the value of ε used for Pokémon dataset

Our work has managed to identify most *legendary* Pokémon. To put it mildly, legendary Pokémon are considered to be the strongest Pokémon in the franchise. We attribute this advantage to the skyline operator. Because of the non-existent overlap between MCSCs, the cliques group skyline Pokémon predominantly based on their types and strength.

Not all Pokémon are easy to come by, specifically the legendary Pokémon. Therefore, players have to make do with other Pokémon hoping their alternative Pokémon offer comparative performance. We attempt to evaluate our MCSCs as good alternative recommenders. To do this, we use a Pokémon battle predictor that learns from a large dataset of battles[Challenge, 2017]. Given two Pokémon, the predictor predicts the winner of the battle. We show that the outcomes of using competitive Pokémon are almost the same. In Fig. 7.10, we visualize a matrix with a set of Pokémon along the rows and columns. A -1 red cell ($+1$ blue cell) indicates that the row Pokémon typically loses (wins) against the column Pokémon. The rows have been grouped into MCSCs (except the last group of non-competitive Pokémon). We observe that in most cases, the performance of Pokémon within the same MCSC are similar. The existence of the similarities is evidenced when contrasted against the last group of non-competitive Pokémon rows. However, there are exceptions: for instance, the performance of Mega Garchomp and Zekrom are different; this is because Zekrom is a legendary Pokémon and, as a result, is a lot stronger.

MCSC ID	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
1	Mega Venusaur	Grass	Poison	80	100	123	122	120	80	1	FALSE
2	Mega Charizard X	Fire	Dragon	78	130	111	130	85	100	1	FALSE
	Mega Altaria	Dragon	Fairy	75	110	110	110	105	80	3	FALSE
3	Arcanine	Fire		90	110	80	100	80	95	1	FALSE
	Simisear	Fire		75	98	63	98	63	101	5	FALSE
4	Rapidash	Fire		65	100	70	80	80	105	1	FALSE
	Simisear	Fire		75	98	63	98	63	101	5	FALSE
5	Muk	Poison		105	105	75	65	100	50	1	FALSE
	Amoonguss	Grass	Poison	114	85	70	85	80	30	5	FALSE
6	Articuno	Ice	Flying	90	85	100	95	125	85	1	TRUE
7	Mega Mewtwo X	Psychic	Fighting	106	190	100	154	100	130	1	TRUE
8	Mega Steelix	Steel	Ground	75	125	230	55	95	30	2	FALSE
	Mega Aggron	Steel		70	140	230	60	80	50	3	FALSE
9	Mega Houndoom	Dark	Fire	75	90	90	140	90	115	2	FALSE
	Darkrai	Dark		70	90	90	135	90	125	4	TRUE
10	Raikou	Electric		90	85	75	115	100	115	2	TRUE
	Mega Manectric	Electric		70	75	80	135	80	135	3	FALSE
11	Suicune	Water		100	75	115	90	115	85	2	TRUE
12	Hariyama	Fighting		144	120	60	40	60	50	3	FALSE
13	Mega Sharpedo	Water	Dark	70	140	70	110	65	105	3	FALSE
	Mega Absol	Dark		65	150	60	115	60	115	3	FALSE
14	Mega Latios	Dragon	Psychic	80	130	100	160	120	110	3	TRUE
	Dialga	Steel	Dragon	100	120	120	150	100	90	4	TRUE
	Palkia	Water	Dragon	90	120	100	150	120	100	4	TRUE
	Reshiram	Dragon	Fire	100	120	100	150	120	90	5	TRUE
15	Mega Garchomp	Dragon	Ground	108	170	115	120	95	92	4	FALSE
	Zekrom	Dragon	Electric	100	150	120	120	100	90	5	TRUE
16	Leafeon	Grass		65	110	130	60	65	95	4	FALSE
	Gourgeist Average Size	Ghost	Grass	65	90	122	58	75	84	6	FALSE
17	Heatran	Fire	Steel	91	90	106	130	106	77	4	TRUE
18	Regigigas	Normal		110	160	110	80	110	100	4	TRUE
19	Giratina Altered Forme	Ghost	Dragon	150	100	120	100	120	90	4	TRUE
	Giratina Origin Forme	Ghost	Dragon	150	120	100	120	100	90	4	TRUE
20	Serperior	Grass		75	75	95	75	95	113	5	FALSE
	Whimsicott	Grass	Fairy	60	67	85	77	75	116	5	FALSE
21	Leavanny	Bug	Grass	75	103	80	70	80	92	5	FALSE
	Scolipede	Bug	Poison	60	100	89	55	69	112	5	FALSE
22	Zoroark	Dark		60	105	60	120	60	105	5	FALSE
	Greninja	Water	Dark	72	95	67	103	71	122	6	FALSE
23	Diancie	Rock	Fairy	50	100	150	100	150	50	6	TRUE

Figure 7.9: Sample MCSCs of the Pokémon dataset

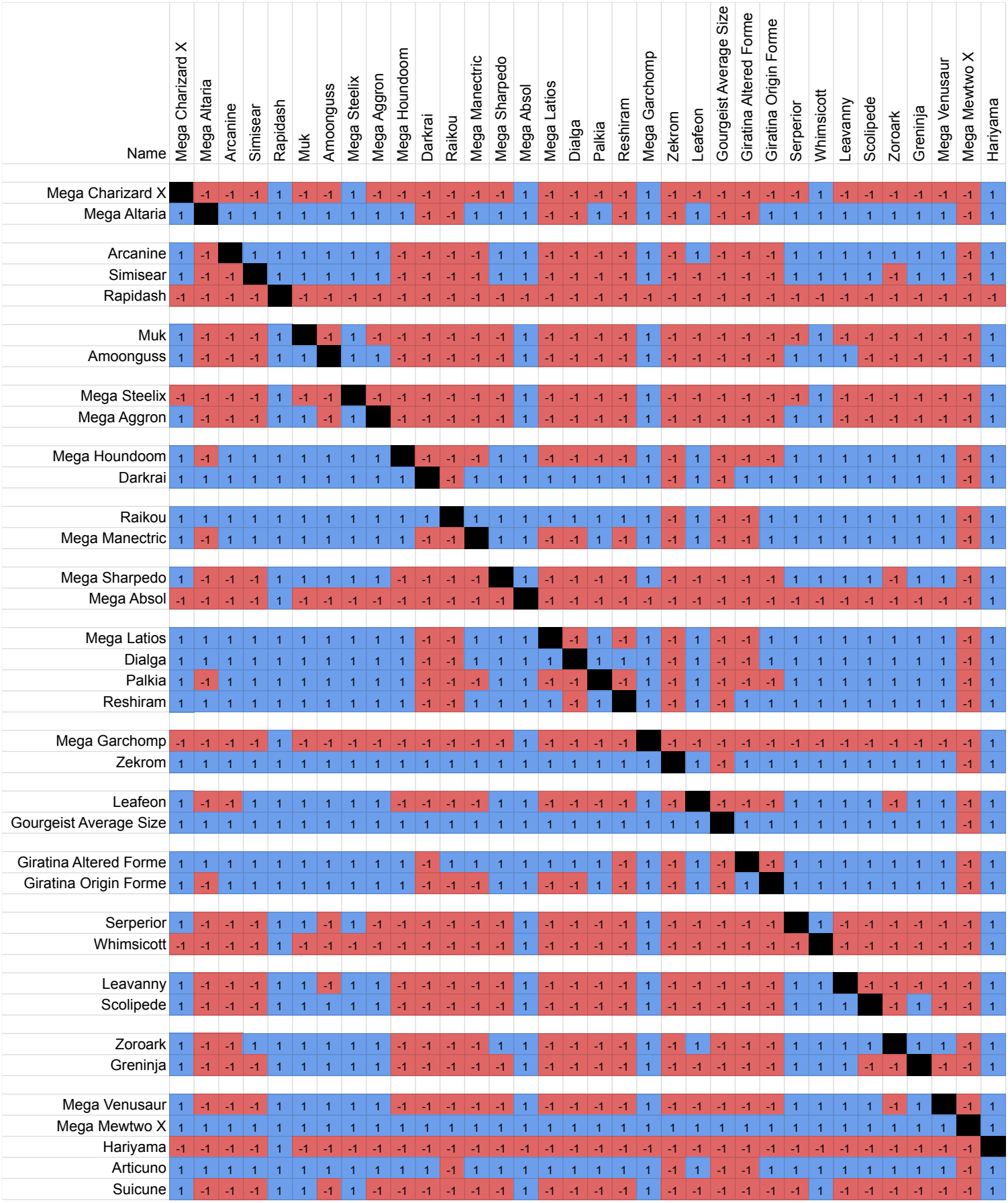
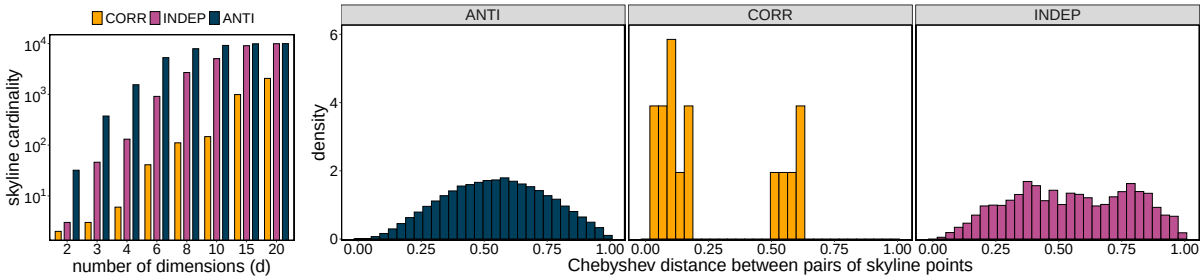


Figure 7.10: Sample MCSCs of the Pokémon dataset. The matrix consists of skyline Pokémon on the rows and columns; A red (blue) cell indicates that the row Pokémon loses (wins) against the column Pokémon. All grouped rows save the last group are MCSCs

7.4 Synthetic Datasets

Experiments are performed using synthetic datasets generated as specified by [Borzsony et al., 2001] using the generator built by [Eder, 2008]. Three types of datasets are generated — (1) CORR - all attributes are correlated, (2) INDEP - all attributes are independent and (3) ANTI - points that are good in one attribute are bad in at least one other attribute. We sample cardinalities from the range $[100, 1M]$, dimensionalities from range $[2, 20]$ and ε from $[0.1\%, 20\%]$. Unless otherwise specified, the default dataset used is of type ANTI with four attributes, 10K points and threshold 5%.



(a) (b)

Figure 7.11: Properties of skyline set of synthetic datasets – (a) distribution of skyline cardinalities when $n = 10K$ and (2) distribution of Chebyshev distances between pairs of skyline points when $d = 4$ and $n = 10K$.

MCSCs are dependent on the properties of the skyline set. Fig. 7.11 depicts properties of the skyline set of the generated synthetic datasets. Fig. 7.11a plots the number of skyline points in datasets with $n = 10K$ as the number of dimensions increases. As described in [Borzsony et al., 2001], the size of the skyline set grows with the dimensionality of the dataset. Additionally, ANTI datasets generally have larger skyline sets than INDEP datasets, and INDEP datasets generally have larger skyline sets than CORR datasets. Notice that as the dimensionality increases, the sparsity of the dataset increases as well, causing the saturation of the skyline set to the whole dataset. Fig. 7.11b plots the distribution of Chebyshev distances between pairs of skyline points of the three dataset types. Note that skyline points of the CORR datasets tend to be close, unlike the ANTI or INDEP datasets. The skyline sets of the INDEP and ANTI datasets are relatively sparse, with high inter-point distances.

The experimental results are summarised in Figs. 7.12 to 7.15. **Unless otherwise specified, the cardinality of the datasets is 10K, the dimensionality is four and threshold $\varepsilon = 0.05$.** We report the run time of our algorithms on these generated datasets in Fig. 7.15; we average the time taken over 10 iterations of the algorithms.

7.4.1 Number of MCSCs

Fig. 7.12 measures the effect of varying number of dimensions (Fig. 7.12a), cardinality (Fig. 7.12b) and threshold (Fig. 7.12c) on the number of MCSCs generated.

1. As the number of dimensions increase, the skyline sets tend to get increasingly sparse. When $d = 20$, the MCSCs converge to non-overlapping singleton skyline point sets.

2. As the number of points increase, the density of the skyline set tends to increase. New competitive relations are introduced, leading to more and larger MCSCs. This result is corroborated in Fig. 7.13b.

3. The effect of increasing the threshold ε is split across datasets — the number of MCSCs tend to decrease in CORR datasets since CORR skyline points tend to be densely packed; an increase in ε causes smaller MCSCs to merge creating larger and fewer MCSCs. However, this is not the case with the ANTI dataset; ANTI skyline points are sparse; an increase in ε introduces more competitive relationships; however, ε will have to be considerably large for small MCSCs to merge. Instead, MCSCs tend to grow in size, and new MCSCs are generated.

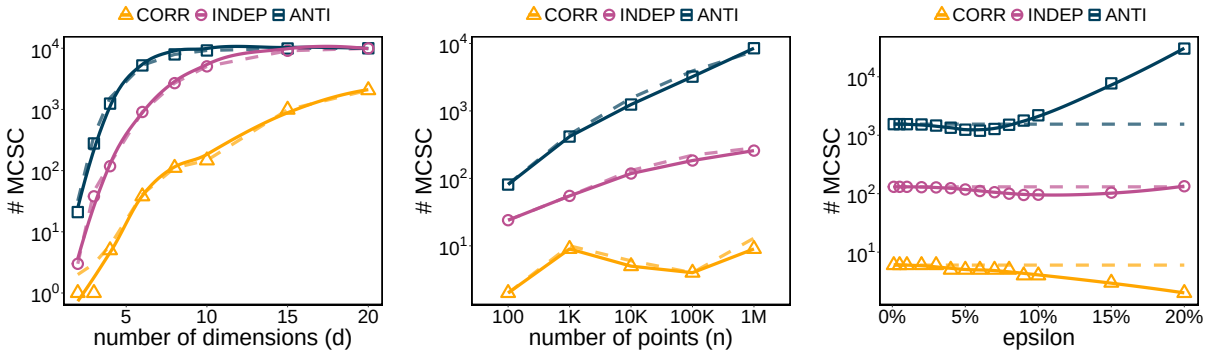


Figure 7.12: Number of MCSCs generated when varying (a) the number of dimensions d ($n = 10K, \varepsilon = 0.05$), (b) cardinality n ($d = 4, \varepsilon = 0.05$) and (c) threshold ε ($d = 4, n = 10K$). **The dashed lines represent the size of the skyline set.**

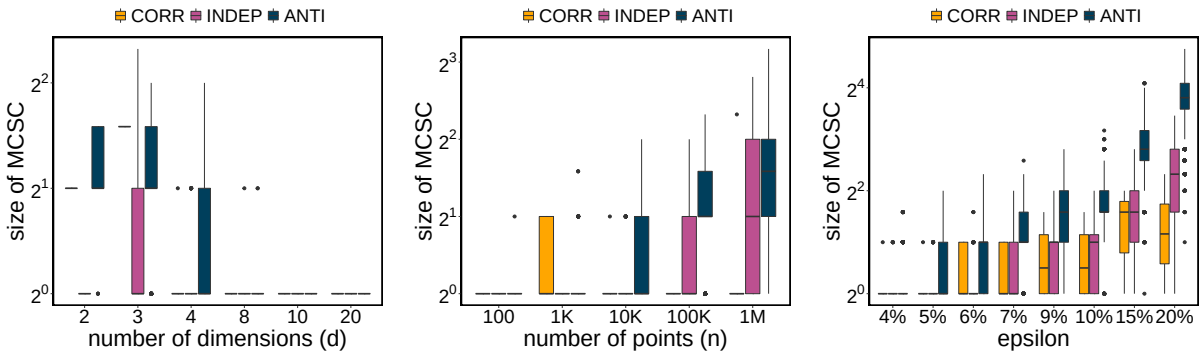
7.4.2 Cardinalities and overlaps of MCSCs

Figs. 7.13 and 7.14 measures the effect of varying number of dimensions (Figs. 7.13a and 7.14a), cardinality (Figs. 7.13b and 7.14b) and threshold (Figs. 7.13c and 7.14c) on the cardinalities and overlaps of the generated MCSCs respectively.

1. As the number of dimensions increase, the skyline sets tend to get increasingly sparse, creating smaller less-overlapping MCSCs converging to singleton cliques.

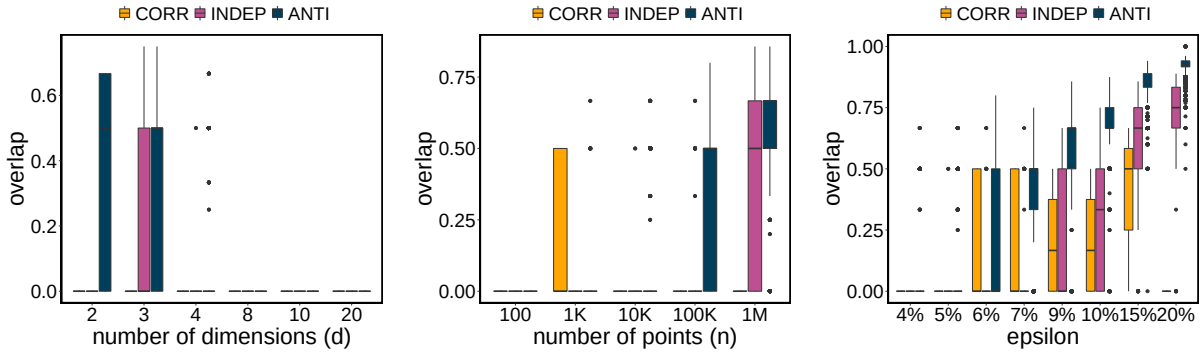
2. As the number of points increases, the density of the skyline set and consequently the cardinalities of MCSCs increase. This also consequently increases overlaps between MCSCs.

3. As is expected, the cardinality and overlaps of MCSCs increase with the threshold ε. This is a natural result – if two skyline points are competitive when $\varepsilon = \varepsilon'$, then they remain competitive for all values of $\varepsilon > \varepsilon'$.



(a) (b) (c)

Figure 7.13: Size of MCSCs or number of points in MCSCs when varying (a) the number of dimensions d ($n = 10K, \varepsilon = 0.05$), (b) cardinality n ($d = 4, \varepsilon = 0.05$) and (c) threshold ε ($d = 4, n = 10K$).

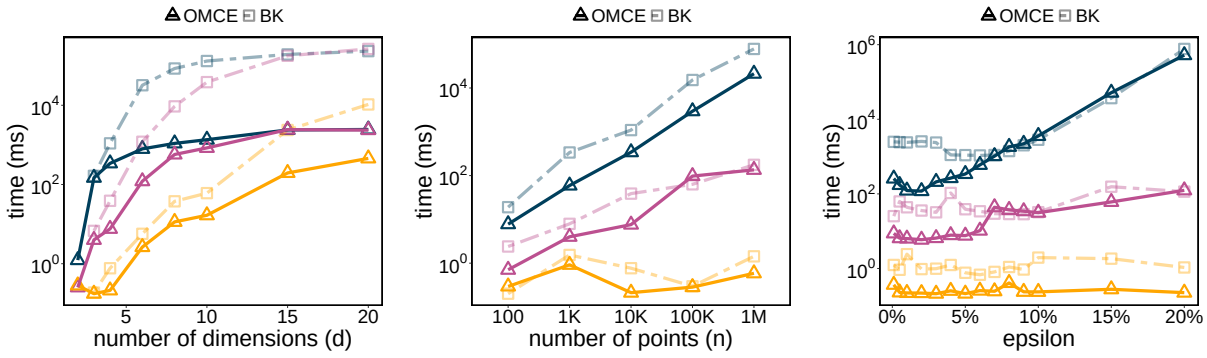


(a) (b) (c)

Figure 7.14: Overlap of MCSCs when varying (a) the number of dimensions d ($n = 10K, \varepsilon = 0.05$), (b) cardinality n ($d = 4, \varepsilon = 0.05$) and (c) threshold ε ($d = 4, n = 10K$).

7.4.3 Performance of OMCE algorithm

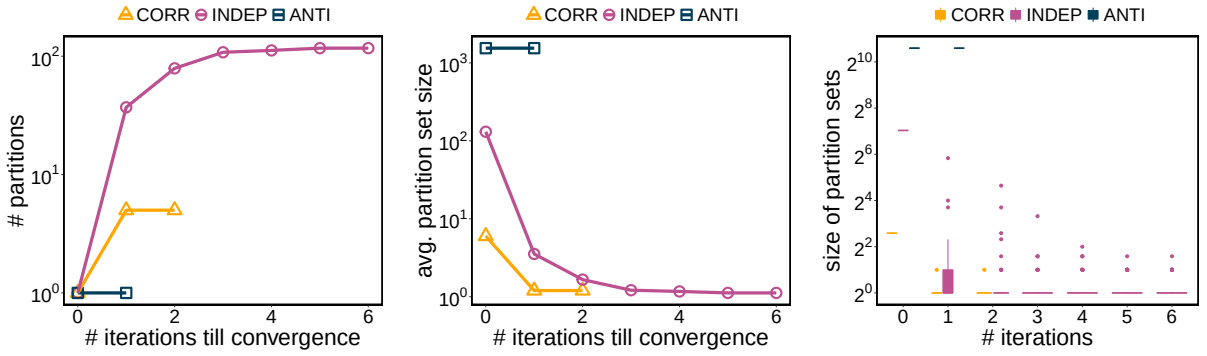
Fig. 7.15 measures the effect of varying number of dimensions (Fig. 7.15a), cardinality (Fig. 7.15b) and threshold (Fig. 7.15c) on the performance of the OMCE algorithm. Furthermore, the plots also compare the performance of our OMCE algorithm against the popular maximal clique enumeration algorithm titled the Bron-Kerbosch algorithm (BK) proposed by [Bron and Kerbosch, 1973]. This algorithm is known to be theoretically optimal in the worst case, i.e. its run time complexity is $\mathcal{O}(3^{n/3})$ where n is the number of vertices. The run time of the BK algorithm is shown Fig. 7.15 in faded colours. The OMCE algorithm tends to perform well on sparse comp graphs. Therefore, the OMCE algorithm outperforms the BK algorithm when the number of dimensions is large. In other cases, the performance of the OMCE algorithm is comparable to that of the BK algorithm. Furthermore, the plots in Fig. 7.15 resemble those of Fig. 7.12, and the performance of the OMCE algorithm depends on the number of MCSCs.



(a) (b) (c)

Figure 7.15: Performance of the OMCE algorithm when varying (a) the number of dimensions d ($n = 10K, \varepsilon = 0.05$), (b) cardinality n ($d = 4, \varepsilon = 0.05$) and (c) threshold ε ($d = 4, n = 10K$). **In each plot, the translucent or faded line plots indicate the performance of the BK algorithm for the same configurations.**

7.4.4 Partitioning



(a) (b) (c)

Figure 7.16: Properties of skyline partition sets: We plot the (a) number of partition sets, (b) the average size of each partition set and (c) the distribution of partition set sizes as the partition algorithm is run multiple times till convergence. The synthetic datasets have configuration cardinality $n = 10K$, dimensionality $d = 4$ and threshold $\varepsilon = 0.05$.

In Section 5.1, we introduce the partitioning algorithm as a preprocessing step. In this section, we empirically demonstrate that performing many iterations of the partitioning algorithm is not necessary and that convergence occurs after a few iterations. We ran the partitioning algorithm on all synthetic dataset configurations until convergence. We observed that the maximum number of iterations was 6 and that the average number of iterations necessary for convergence is 1.7857. Figs. 7.16a to 7.16c plot the number of partition sets, the average size of each partition set and the distribution of sizes of partition sets, respectively as the partition algorithm is applied multiple times. The partition algorithm is

not always effective; in both the INDEP and CORR cases, the partition algorithm effectively partitions the dataset but not in the ANTI case. The partition algorithm is most effective when the threshold ε is small.

Chapter 8

2D Competitive Skyline Cliques

We treat the 2D case differently because of an interesting property exhibited by 2D skyline sets. In this section, we assume that the two dimensions (or attributes) are a_1 and a_2 and that the points in the skyline set $\mathcal{S} = \{p_1, p_2, \ldots, p_m\}$ are implicitly sorted by the first attribute a_1 in ascending order as is the case for the hotels dataset (Fig. 1.1).

Property 8.1. If $\mathcal{S}$ is a 2D skyline set and the points in $\mathcal{S}$ are sorted in ascending order based on one attribute (say a_1), then they are simultaneously sorted in descending order by the other attribute (a_2)

Proof. Although this property has been proposed by [Borzsony et al., 2001], we provide a proof for completeness. We use a proof by contradiction. Let p_i and p_j be two skyline points such that $i < j$. This implies that $p_i[a_1] \leq p_j[a_1]$ (skyline set is sorted) and $p_i[a_2] < p_j[a_2]$.

$$\begin{aligned} p_i[a_1] \leq p_j[a_1] \wedge p_i[a_2] < p_j[a_2] &\implies p_i \preceq p_j \\ &\implies p_j \text{ is not a skyline point} \qquad \text{(contradiction)} \end{aligned}$$

□

This property has two important consequences: Corollary 8.1 and Corollary 8.2

Corollary 8.1. If 2D skyline points are sorted by some attribute (say a_1) and p_i and p_j are two skyline points such that $i < j$, then

1. If p_i is not competitive with p_j then for all integers $k \geq j$, p_i is not competitive with p_k

$$\neg comp(p_i, p_j) \implies \forall_{k \geq j} \neg comp(p_i, p_k)$$

2. If p_i is competitive with p_j then for all integers $i \leq k \leq j$, p_i is competitive with p_k

$$comp(p_i, p_j) \implies \forall_{i \leq k \leq j} comp(p_i, p_k)$$

Proof. We first prove the former statement: Let p_i and p_j be two skyline points such that $i < j$ and p_i is not competitive with p_j. Then,

$$\begin{aligned} i < j \wedge \neg comp(p_i, p_j) &\implies max(p_j[a_1] - p_i[a_1], p_i[a_2] - p_j[a_2]) > \varepsilon \\ &\implies \forall_{k \geq j}\ max(p_k[a_1] - p_i[a_1], p_i[a_2] - p_k[a_2]) > \varepsilon \\ &\qquad (\forall_{k \geq j}\ p_k[a_1] > p_j[a_1] \wedge p_j[a_2] > p_k[a_2]) \\ &\implies \forall_{k \geq j}\ \neg comp(p_i, p_k) \end{aligned}$$

We now prove the latter statement: Let p_i and p_j be two skyline points such that $i < j$ and p_i is competitive with p_j. Then,

$$\begin{aligned} i < j \wedge comp(p_i, p_j) &\implies max(p_j[a_1] - p_i[a_1], p_i[a_2] - p_j[a_2]) \leq \varepsilon \\ &\implies \forall_{i \leq k \leq j}\ max(p_k[a_1] - p_i[a_1], p_i[a_2] - p_k[a_2]) \leq \varepsilon \\ &\qquad (\forall_{i \leq k \leq j}\ p_i[a_1] \leq p_k[a_1] \leq p_j[a_1] \wedge p_j[a_2] \leq p_k[a_2] \leq p_i[a_2]) \\ &\implies \forall_{i \leq k \leq j}\ comp(p_i, p_k) \end{aligned}$$

□

Corollary 8.2. The mbb of a set of 2D skyline points is characterised by at most two points in the set. If $\mathbf{s}$ is a set of 2D skyline points and if $p_{min}, p_{max} \in \mathbf{s}$ are the points with the smallest and largest values of the first dimension a_1 in $\mathbf{s}$ respectively, then $mbb(\mathbf{s})$ is given by

$$\begin{aligned} mbb(\mathbf{s}).lb &= (p_{min}[a_1],\ p_{max}[a_2]) \\ mbb(\mathbf{s}).ub &= (p_{max}[a_1],\ p_{min}[a_2]) \end{aligned}$$

Proof. Let $\mathbf{s} = \{p_1, \ldots, p_k\}$ be a sorted set of 2D skyline points sorted by the first dimension (a_1) $p_1[a_1] \leq p_2[a_1] \leq \ldots \leq p_k[a_1]$. Then, according to Property 8.1, $\mathbf{s}$ is also sorted by the second dimension but in descending order $p_1[a_2] \geq p_2[a_2] \geq \ldots \geq p_k[a_2]$. Then the mbb (defined in Section 5.2) is given by the following equations

$$\begin{aligned} mbb(\mathbf{s}).lb[a_1] &= \min_{p \in \mathbf{s}}\ p[a_1] = p_1[a_1] & mbb(\mathbf{s}).lb[a_2] &= \min_{p \in \mathbf{s}}\ p[a_1] = p_k[a_1] \\ mbb(\mathbf{s}).ub[a_1] &= \max_{p \in \mathbf{s}}\ p[a_1] = p_k[a_1] & mbb(\mathbf{s}).ub[a_2] &= \max_{p \in \mathbf{s}}\ p[a_1] = p_1[a_1] \end{aligned}$$

□

An implication of the two corollaries is that there are at most $|\mathcal{S}|$ MCSCs. 2D bounding boxes of subsets of the skyline set are characterised by 2 points (Corollary 8.2). If point p is chosen as the point having the minimum value of dimension a_1 (and consequently the maximum value of the other dimension a_2) in the MCSC being constructed, then the MCSC is given by the set of all skyline points in box bb where bb is defined by the following equations (Lemma 6.1)

$$\begin{aligned} bb.lb &= (p[a_1], \quad p[a_2] - \varepsilon) \\ bb.ub &= (p[a_1] + \varepsilon, \; p[a_2] \quad) \end{aligned}$$

While the worst-case number of CSCs of the 2D skyline remains the same, the properties exhibited by 2D MCSCs allow for a faster and optimal enumeration algorithm. We present an $\mathcal{O}(n)$ time sweep-line algorithm (Algorithm 3) that enumerates all 2D MCSCs. The sweepline algorithm assumes that the skyline set has already been computed and sorted, without loss of generality, by the first dimension a_1. This assumption is practical since the optimal $\mathcal{O}(n)$ time 2D skyline computation algorithm[Borzsony et al., 2001] generates the skyline set sorted by the first attribute.

Algorithm 3 2D Sweepline **M**aximal **C**ompetitive Skyline Clique **E**numeration Algorithm (2DSMCE)

1: **Input:** $\mathcal{S} = \{p_1, p_2, \ldots, p_m\}$ sorted by a_1, ε
2: $\mathcal{M} \longleftarrow \phi$; $p_{min} \longleftarrow p_1$; $p_{max} \longleftarrow p_1$
3: **while** $p_{min} \in \mathcal{S}$ and $p_{max} \in \mathcal{S}$ **do**
4: **while** $\neg comp(p_{min}, p_{max})$ **do**
5: $p_{min} \longleftarrow p_{min+1}$
6: **while** $p_{max+1} \in \mathcal{S}$ and $comp(p_{min}, p_{max+1})$ **do**
7: $p_{max} \longleftarrow p_{max+1}$
8: $\mathbf{m} \longleftarrow \{p_i \mid min \leq i \leq max\}$
9: $\mathcal{M} \longleftarrow \mathcal{M} \cup \{\mathbf{mcc}\}$
10: $p_{max} \longleftarrow p_{max+1}$
11: **Return:** $\mathcal{M}$

The sweep-line algorithm work by moving a variable-sized window through the sorted skyline set. Points p_{min} and p_{max} define the smallest and largest values of attribute a_1 in the MCSC being constructed. Each value of p_{min} results in one MCSC. The loop defined in lines 4–5 generates a viable p_{min}, lines 6–7 determine the corresponding value of p_{max} (Corollary 5.2) and line 8 construct the new MCSC. Note that not every p_{min} generates a MCSC; for instance, if $p_{min} = p_3$, the generated clique is 3, which is not maximal because it is contained in 123. To circumvent this, we use line 10 to ensure the constructed clique is not contained in any other maximal clique.

Example. The hotels dataset is sorted by attribute Price. The algorithm starts by processing p_1 in line 1. Lines 6–7 compute the appropriate p_{max} which is p_1. MCSC 1 is included in M. p_{max} is updated to p_2. Lines 4-5 updates p_{min} to p_2 and consequently lines 6–7 update p_{max} to p_3. MCSC 23 is added to $\mathcal{M}$. Line 10 updates p_{max} to p_4. Lines 4–5 update p_{min} to p_3 and lines 6–7 consequently update p_{max} to p_5. MCSC 345 is included in $\mathcal{M}$. The run of the algorithm is demonstrated in *Fig.* 8.1.

Point	1	2	3	4	5	6	7	8
Price	0.15	0.16	0.17	0.18	0.19	0.20	0.23	0.25
Distance	0.40	0.34	0.31	0.28	0.26	0.20	0.185	0.17

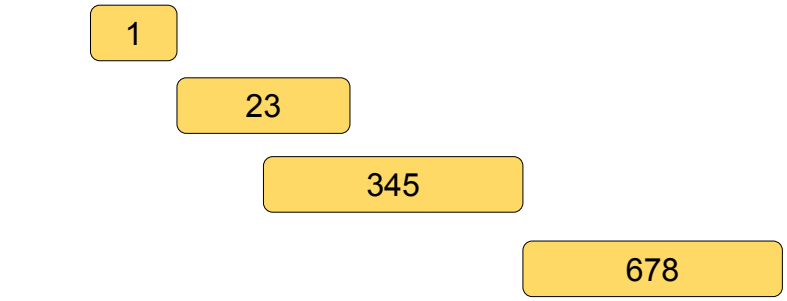
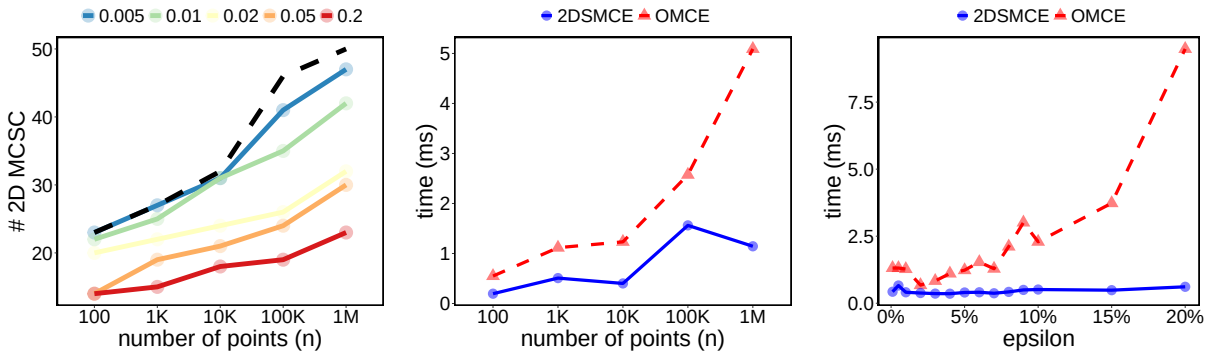


Figure 8.1: Run of Algorithm 3 on hotels dataset.

Experimental Analysis

In Fig. 8.2a, we plot the number of 2DMCSCs generated as the cardinality of the underlying dataset is varied. We create plots for multiple values of ε. The black dashed line represents the cardinality of the skyline set. This plot empirically verifies our claim: there are at most $|\mathcal{S}|$ 2DMCSCs. In Figs. 8.2b and 8.2c, we compare the performance of the 2DSMCE, and the OMCE algorithms as the dataset cardinality and threshold are varied, respectively. As the cardinality of the dataset increases, so does the cardinality of the skyline set, consequently, the runtime of the 2DSMCE algorithm. On the other hand, the performance of the 2DSMCE algorithm is independent of the threshold ε. In both cases, the 2DSMCE outperforms the OMCE algorithm.



(a) (b) (c)

Figure 8.2: Experimental results on 2D ANTI datasets. (a) The number of 2D MCSCs generated as both cardinality and threshold are varied. The black dashed line represents the cardinality of the skyline set. (b),(c) Performance of 2DSMCE in comparision to the OMCE algorithm as the cardinality ($\varepsilon = 0.05$) and threshold ($n = 10K$) are varied respectively.

Chapter 9

Approximate Competitive Skyline Cliques

The number of MCSCs can be large, and enumerating MCSCs can be expensive. We propose an approximation that consists of $\mathcal{O}(|\mathcal{S}|)$ cliques and can be computed in $\mathcal{O}(|\mathcal{S}|^2)$ time.

Definition 9.1

An **approximate competitive skyline clique** (ACSC for short) consists of a skyline point p and all of its competitors (ε-neighborhood). $\mathcal{AC}$ denotes the set of all ACSCs.

$$\mathcal{AC} = \{competitors(p, \mathcal{S}) \mid p \in \mathcal{S}\}$$

The approximation is bounded by Theorem 9.1.

Theorem 9.1

Let p be a skyline point, then the Chebyshev distance between any two points in set $competitors(p, \mathcal{S})$ is at most 2ε

Proof. The proof relies on the transitive bound property (see proof of Corollary 5.3): If p, q, r be three skyline points such that $comp(p, q)$ and $comp(q, r)$, then $||p - r||_\infty \leq 2\varepsilon$. Let p be a skyline point and let q_1 and q_2 be points in $competitors(p, \mathcal{S})$, then $comp(q_1, p)$ and $comp(p, q_2)$. By the transitive bound property, $||q_1 - q_2||_\infty \leq 2\varepsilon$. □

Example. For example, the ACSCs of the hotels dataset are

$$\mathcal{AC} = \{1, 23, 2345, 345, 678\}$$

For the hotels dataset, $\varepsilon = 0.05 = 5\%$. Therefore, any two points in any ACSC differ by at most 10% across any attribute.

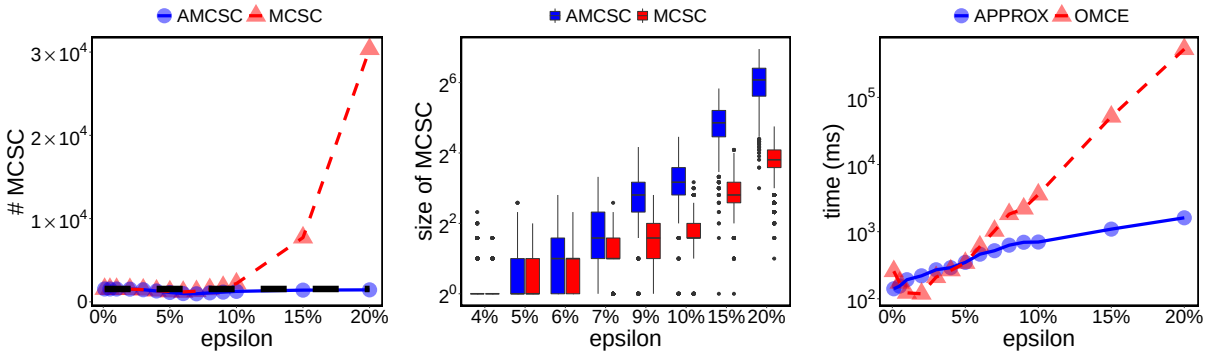
As an optimization, subsumed ACSCs like 23 can be eliminated (called **approximate maximal competitive cliques** (AMCSC)). The set of all AMCSCs is denoted by $\mathcal{AMC}$.

$$\mathcal{AMC} = \{1, 2345, 678\}$$

To enumerate ACSCs, we can iterate through the skyline set and perform a range query for each skyline point. The range query can be implemented efficiently as detailed in Chapter 5.

Experimental Analysis

In this section, we demonstrate the utility and practicality of AMCSCs. Note that in Fig. 7.12c, the number of MCSCs of the ANTI dataset are much larger than the number of skyline points. Specifically, the dataset consists of at most 10K skyline points and over 30K MCSCs. Furthermore, these MCSCs tend to be large in size (Fig. 7.13c) and have considerable overlap (Fig. 7.14c). Hence, many MCSCs tend to be redundant. In situations like this, AMCSCs can yield better results. In Fig. 9.1b, we plot the number of AMCSCs and MCSCs generates as the threshold ε is varied. The black-coloured dashed line represents the cardinality of the skyline set. Note that the number of AMCSCs is consistently smaller than the skyline cardinality and far smaller than the number of MCSCs. In Fig. 9.1b, we plot the distribution of MCSC and AMCSC cardinalities as the threshold is varied and in Fig. 9.1c we plot the time taken to compute AMCSCs and MCSCs as the threshold is varied. As is expected, AMCSCs tend to be larger than MCSC. However, AMCSCs are far more efficient to enumerate.



(a) (b) (c)

Figure 9.1: Experimental results on AMCSCs of 4D ANTI dataset with 10K points. (a) The number of AMCSCs and MCSCs generated as the threshold is varied. **The black dashed line represents the cardinality of the skyline set.** (b) Compares cardinalities or sizes of AMCSCs and MCSCs as the threshold is varied. (c) Compares the efficiency of enumerating all AMCSCs and MCSCs as the threshold is varied.

Chapter 10

Conclusion

In this work, we formalised the notion of competitiveness between skyline points and defined competitive skyline cliques (CSCs) and maximal competitive skyline cliques (MCSCs). We restricted our attention to MCSCs because many CSCs can be redundant and CSCs are exponential in number. Because the expected number of MCSCs is low, we employed an output-sensitive algorithm to enumerate all MCSCs. We provided optimisationsto our algorithm that efficiently exploited properties of our competitiveness metric. In our empirical analysis, we showed that the use of an output-sensitive algorithm over another popular maximal clique enumeration algorithm (BK) witnessed a significant performance improvement. We proved that, unlike the problem of maximal clique enumeration, the problem of enumerating all MCSCs is **not** NP-Hard and can, in fact, be solved in polynomial time. We established this fact by providing a polynomial bound on the number of MCSCs. Empirically, we observed that in most cases, the number of MCSCs was about the same as the number of skyline points. However, within real datasets, we observed that the number of MCSCs was far fewer than the number of skyline points. We demonstrated the utility of MCSCs using three real-world datasets. With the UCars dataset, we observed that MCSCs generated sets of used cars that targetted different consumer sets. We observed sets of cars that differed in the economic and purpose-driven preferences a consumer might have. For instance, purpose-driven vehicles like the TATA 14-seater formed singleton MCSCs where economic vehicles like Maruti Swift and Ford Figo appeared in the same clique. With the FIFA22 dataset, we observed how MCSCs can be utilized within constraints — A player is competitive with another player only if they play one of the same positions. We evaluated MCSCs of the Pokémon dataset: we observed that all Pokémon in a MCSC generally tended to win and lose against the same Pokémon. We also showed the existence of an optimal linear-time MCSC enumeration algorithm for the specific case when $d = 2$ that exploited unique properties of 2D skyline points. Finally, we acknowledged that in some cases, the number of MCSCs can be large — to that end, we provided a bounded approximate alternative to MCSCs (AMCSCs) that were smaller in number and easier to enumerate.

Choosing the threshold ε can be tricky. Currently, we use two approaches — (1) in the first approach, we first set ε for one attribute (for instance, Price) and then reason how much of a change we expect in another attribute for an ε change in the fixed attribute and (2) in the second approach, we query the user

with pairs of products to generate an annotated dataset; we then determine ε that satisfies most of the annotations. A problem with picking a strict value of ε is the lack of fuzziness within competitiveness relationships. In future work, we will attempt to develop a notion of competitiveness that is not as strict. MCSCs have other applications – for instance, these cliques are capable of identifying competitive market segments. There has been work that aimed at generating new skyline products [Wan et al., 2009]. However, not all skyline products are attractive to consumers. Competitive market segments could be construed as good indicators of popular consumer preferences. A possible future application would be to build skyline products that are competitive with many other skyline products to increase the attractiveness of the product. Or alternatively, if the objective is to build a diverse set of products, then generating skyline products that are not competitive with any existing skyline product would be a good strategy. Another interesting new direction is generating cliques of diverse skyline points where two skyline points are defined to be diverse if they differ by **at least** ε, a large user-defined threshold ε.

Bibliography

[Agrawal et al., 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Citeseer. 11

[Barradas, 2016] Barradas, A. (2016). Pokemon with stats. https://www.kaggle.com/datasets/abcsds/pokemon. 36

[Birla, 2021] Birla, N. (2021). Vehicle dataset. https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho. 30

[Borzsony et al., 2001] Borzsony, S., Kossmann, D., and Stocker, K. (2001). The Skyline operator. In *Proceedings 17th International Conference on Data Engineering*, pages 421–430, Heidelberg, Germany. IEEE Comput. Soc. 2, 3, 11, 40, 45, 47

[Bron and Kerbosch, 1973] Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577. 12, 42

[Burdick et al., 2005] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., and Yiu, T. (2005). Mafia: A maximal frequent itemset algorithm. *IEEE transactions on knowledge and data engineering*, 17(11):1490–1504. 11

[Challenge, 2017] Challenge, T. H. (2017). Pokemon- weedle's cave. https://www.kaggle.com/datasets/terminus7/pokemon-challenge. 37

[Chomicki et al., 2003] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2003). Skyline with presorting. In *ICDE*, volume 3, pages 717–719. 11

[Eder, 2008] Eder, H. (2008). randdataset. https://www.postgresql.org/ftp/projects/pgFoundry/randdataset/. 40

[Gouda and Zaki, 2005] Gouda, K. and Zaki, M. J. (2005). Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery*, 11(3):223–242. 11

[Greifeneder et al., 2010] Greifeneder, R., Scheibehenne, B., and Kleber, N. (2010). Less may be more when choosing is difficult: Choice complexity and too much choice. *Acta Psychologica*, 133(1):45–50. 1, 4

[Iyengar and Lepper, 2001] Iyengar, S. and Lepper, M. (2001). When Choice is Demotivating: Can One Desire Too Much of a Good Thing? *Journal of personality and social psychology*, 79:995–1006. 1, 4

[Kossmann et al., 2002] Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 275–286. Elsevier. 11

[Leone, 2021] Leone, S. (2021). Fifa 22 complete player dataset. https://www.kaggle.com/datasets/stefanoleone992/fifa-22-complete-player-dataset. 33

[Liu et al., 2015] Liu, J., Xiong, L., Pei, J., Luo, J., and Zhang, H. (2015). Finding pareto optimal groups: Group-based skyline. *Proc. VLDB Endow.*, 8(13):2086–2097. 27

[Makino and Uno, 2004] Makino, K. and Uno, T. (2004). New algorithms for enumerating all maximal cliques. In *Scandinavian workshop on algorithm theory*, pages 260–272. Springer. 12

[Moon and Moser, 1965] Moon, J. W. and Moser, L. (1965). On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28. 25

[Papadias et al., 2003a] Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2003a). An optimal and progressive algorithm for skyline queries. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 467–478. 3, 11

[Papadias et al., 2003b] Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2003b). An optimal and progressive algorithm for skyline queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 467–478, New York, NY, USA. Association for Computing Machinery. 13

[Roberts, 1969] Roberts, F. S. (1969). On the boxicity and cubicity of a graph. *Recent progress in combinatorics*, 1(1):301–310. 26

[Sade and Cohen, 2020] Sade, L. and Cohen, S. (2020). Diverse enumeration of maximal cliques. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3321–3324. 12

[Scheibehenne, 2008] Scheibehenne, B. (2008). The effect of having too much choice. 4

[Spinrad, 2003] Spinrad, J. P. (2003). *Efficient Graph Representations.: The Fields Institute for Research in Mathematical Sciences.*, volume 19. American Mathematical Soc. 26

[Tan et al., 2001] Tan, K.-L., Eng, P.-K., Ooi, B. C., et al. (2001). Efficient progressive skyline computation. In *VLDB*, volume 1, pages 301–310. 11

[Tao, 2006] Tao, Y. (2006). SUBSKY: Efficient computation of skylines in subspaces. In *In ICDE*, page 65. 2

[Tsukiyama et al., 1977] Tsukiyama, S., Ide, M., Ariyoshi, H., and Shirakawa, I. (1977). A New Algorithm for Generating All the Maximal Independent Sets. *SIAM Journal on Computing*, 6(3):505–517. 13, 17

[Uno et al., 2003] Uno, T., Asai, T., Uchida, Y., and Arimura, H. (2003). Lcm: An efficient algorithm for enumerating frequent closed item sets. In *Fimi*, volume 90. Citeseer. 11

[Wan et al., 2009] Wan, Q., Wong, R. C.-W., Ilyas, I. F., Özsu, M. T., and Peng, Y. (2009). Creating competitive products. *Proceedings of the VLDB Endowment*, 2(1):898–909. 53

[Wang et al., 2013] Wang, J., Cheng, J., and Fu, A. W.-C. (2013). Redundancy-aware maximal cliques. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 122–130, New York, NY, USA. Association for Computing Machinery. 12, 29

[Zaki et al., 1997] Zaki, M. J., Parthasarathy, S., Ogihara, M., Li, W., et al. (1997). New algorithms for fast discovery of association rules. In *KDD*, volume 97, pages 283–286. 11