Complexity of Some Polyhedral Enumeration Problems

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

Hans Raj Tiwary

Universität des Saarlandes

Saarbrücken
2008

Acknowledgements

First and foremost, I would like to thank my advisor Prof. Raimund Seidel for introducing me to the wonderful area of the theory of polytopes. He was always there when I needed to discuss anything and his quick counterexamples for many of my conjectures were very helpful in getting me off many wrong tracks. I am very grateful to him for allowing me so much freedom in research and to borrow his own words to his PhD advisor John Gilbert "thanks for not making me fulfill all the promises I made, of things I would do, or would not do".

I would also like to thank Prof. Kurt Mehlhorn and Prof. Friedrich Eisenbrand for agreeing to be on my thesis committee.

Many of the results in this thesis are a product of the work I did with Khaled Elbassioni. His enthusiasm for research in general, his receptiveness of my ideas and his expertise were a constant source of motivation and encouragement for me.

Many thanks go to my office mate, colleague, and academic sibling Saurabh. It was a pleasure to work with him on so many problems. Spending countless hours with him discussing philosophy, geometry and everything else under the sun were a nice escape whenever I got overwhelmed by the burden of my thesis.

I was lucky to have Rali, Josi, and Cris as such close friends. Your company made living in Saarbrücken a wonderful experience. I will sorely miss the everyday after-lunch coffee in MPI with you guys.

I would also like the acknowledge the generous fellowship that I received from the Deusche Forschungsgemeinschaft (DFG) for the first three years of my PhD studies.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den 06.11.2008

(Unterschrift)

Kurzfassung

In dieser Arbeit betrachten wir das Problem, die Halbraumdarstellung eines Polytops in seine Eckendarstellung umzuwandeln, — das sogenannte Problem der Eckenaufzählung — sowie viele andere grundlegende und eng verwandte Berechnungsprobleme für Polytope. Das Problem, die Eckendarstellung in die Halbraumdarstellung umzuwandeln — das sogenannte Konvexe-Hüllen Problem — ist äquivalent zum Problem der Eckenaufzählung.

In Kapitel 3 zeigen wir, dass Eckenaufzählung für ein unbeschränktes $\mathcal{H}$-Polyeder P selbst dann NP-schwer ist, wenn P nur 0/1-Ecken hat. Das verbessert ein Ergebnis von Khachiyan et. al. [KBB$^+$06]. In den Kapiteln 4 bis 6 zeigen wir, dass viele andere Operationen auf Polytopen, wie Berechnung von Minkowski-Summe, Durchschnitt, Projektion usw., für viele Darstellungen NP-schwer sind und für viele weitere äquivalent zu Eckenaufzählung sind.

In Kapitel 7 beweisen wir Härteresultate über ein Kegelüberdeckungsproblem, das danach fragt, ob eine gegebene Menge polyedrischer Kegel eine andere gegebene Menge überdeckt. Wir zeigen, dass dies im Allgemeinen ein NP-vollständiges Problem ist und wichtige Probleme wie Eckenaufzählung und Hypergraphentraversierung als Spezialfälle umfasst.

Schließlich stellen wir in Kapitel 8 einen Zusammenhang zwischen Eckenaufzählung und Graphisomorphie her, indem wir beweisen, dass ein bestimmtes Graphisomorphie-schweres Problem genau dann Graphisomorphie-leicht ist, wenn Eckenaufzählung Graphisomorphie-leicht ist. Außerdem beantworten wir eine Frage von Kaibel und Schwartz über das Testen der Selbst-Dualität von Polytopen.

Abstract

In this thesis we consider the problem of converting the halfspace representation of a polytope to its vertex representation — the Vertex Enumeration problem — and various other basic and closely related computational problems about polytopes. The problem of converting the vertex representation to halfspace representation — the Convex Hull problem — is equivalent to vertex enumeration.

In chapter 3 we prove that enumerating the vertices of an unbounded $\mathcal{H}$-polyhedron P is NP-hard even if P has only $0/1$ vertices. This strengthens a previous result of Khachiyan et. al. [KBB$^+$06]. In chapters 4 to 6 we prove that many other operations on polytopes like computing the Minkowski sum, intersection, projection, etc. are NP-hard for many representations and equivalent to vertex enumeration in many others.

In chapter 7 we prove various hardness results about a cone covering problem where one wants to check whether a given set of polyhedral cones cover another given set. We prove that in general this is an NP-complete problem and includes important problems like vertex enumeration and hypergraph transversal as special cases.

Finally, in chapter 8 we relate the complexity of vertex enumeration to graph isomorphism by proving that a certain graph isomorphism hard problem is graph isomorphism easy if and only if vertex enumeration is graph isomorphism easy. We also answer a question of Kaibel and Schwartz about the complexity of checking self-duality of a polytope.

Zusammenfassung

Jedes Polytop $P \subset \mathbb{R}^d$ kann sowohl als konvexe Hülle von endlich vielen Punkten in $\mathbb{R}^d$ als auch als beschränkter Durchschnitt von endlich vielen Halbräumen dargestellt werden. Jede dieser Darstellungen bestimmt die andere eindeutig (falls Mehrdeutigkeiten ausgeschlossen wurden), aber es sind keine output-sensitiven Algorithmen bekannt, die eine dieser Darstellungen in die andere überführen. Je nachdem, ob ein Polytop durch Ecken- oder Facettendarstellung gegeben ist, kann die Berechnungskomplexität einer Operation auf Polytope sehr unterschiedlich sein. Diese Arbeit behandelt verschiedene Berechnungsprobleme für Polytope und ihre Komplexität.

Die Probleme und Ergebnisse dieser Arbeit behandeln folgende gemeinsamen Themen:

- Viele der in dieser Arbeit betrachteten Probleme sind sehr grundlegende Operationen auf Polytopen. Operationen wie Durchschnitt, Minkowski-Addition und Projektion sind Basisoperationen auf Polytopen, die in der Praxis oft berechnet werden müssen. Die Ergebnisse dieser Arbeit zeigen, dass die beste Methode, diese Operationen durchzuführen, oft darin besteht, als erstes die Darstellung des Eingabepolytops umzuwandeln, was die Bedeutung von output-sensitiven Algorithmen zur Eckenaufzählung unterstreicht.

- Jedes in dieser Arbeit behandelte Problem ist eng verwandt mit dem Problem der Eckenaufzählung (vertex enumeration - VE). Für die meisten Probleme würde ein effizienter Algorithmus zu einem output-sensitiven Algorithmus für Eckenaufzählung führen. Unglücklicherweise stellen sich die meisten dieser Probleme als NP- oder $\#P$-schwer heraus. Hingegen stellt das Problem zu überprüfen, ob ein Polytop isomorph zu seinem polarem Dual ist (Kapitel 8), einen Zusammenhang zwischen Eckenaufzählung und dem bekanntem Problem der Graphisomorphie (GI) her. Allerdings ist der in dieser Arbeit hergestellt Zusammenhang etwas schwach.

- Jedes Problem unterstreicht die Bedeutung der Darstellung der Eingabe für Polytop-Berechnungsprobleme. So ist beispielsweise die Berechnung des Eckenzentroids (Kapitel 4) für ein $\mathcal{V}$-polytop trivial, während sie für $\mathcal{H}$-Polytope $\#P$-schwer ist. In ähnlicher Weise ist es leicht, die Facetten des Durchschnitts zweier $\mathcal{H}$-Polytope zu berechnen, doch das Problem ist NP-schwer für fast alle anderen Darstellungen.

Die Hauptbeiträge dieser Arbeit unterteilen sich grob in die in den folgenden Abschnitten ausgeführten Kategorien.

Verbesserung früherer Ergebnisse

In Kapitel 3 zeigen wir, dass Eckenaufzählung eines unbeschränkten $\mathcal{H}$-Polyeders P selbst dann NP-schwer ist, wenn P nur 0/1-Ecken besitzt. Das verbessert ein Ergebnis für allgemeine Polyeder von Khachiyan et. al. [KBB$^+$06]. Für allgemeine $\mathcal{H}$-Polytope ist die Komplexität für Eckenaufzählung unbekannt, aber für ein Polytop, dessen Ecken alle 0/1 sind, gibt es einen output-sensitiven Algorithmus [BL98]. Unser Ergebnis verdeutlicht damit den Unterschied in der Berechnungskomplexität der Eckenaufzählung für Polytope gegenüber der von unbeschränkten Polyedern.

Unsere kleinen, aber sehr entscheidenden Veränderungen im Beweis von [KBB$^+$06] ermöglichen es uns, einige zusätzliche Härteresultate zu gewinnen. So zeigt unsere Methode unter anderem, dass nicht in Polynomialzeit festgestellt werden kann, ob ein $\mathcal{H}$-Polyeder ein 0/1-Polyeder ist, außer $P = NP$. Viele dieser zusätzlichen Resultate aus Kapitel 3 sind schon bekannt, aber unsere Methode erlaubt uns, diese Ergebnisse zu vereinheitlichen. So sind zwar nicht die Resultate selbst ein Fortschritt, der vereinheitlichte Beweis hingegen ist durchaus als Fortschritt zu sehen.

In Kapitel 8 zeigen wir, dass es Graphisomorphie-vollständig ist zu prüfen, ob ein Polytop, gegeben durch $\mathcal{HV}$-Darstellung, kombinatorisch isomorph zu seinem polaren Dual ist. Dies verallgemeinert ein Ergebnis von Kaibel und Swartz [KS03], dass die GI-Vollständigkeit des Test auf kombinatorische Isomorphie zweier $\mathcal{HV}$-Polytope beweist. Unser Ergebnis basiert auf einer einfachen Beobachtung, die die Zerlegbarkeit durch free-joins eines Polytops in kleinere Polytope charakterisiert (Lemma 8.5.1).

Neue Härteresultate

Die wichtigsten neuen Ergebnisse dieser Arbeit beschäftigen sich damit zu zeigen, dass viele Grundoperationen auf Polytopen, wie Berechnung von Durchschnitt, Minkowski-Summe, Projektion usw., im Allgemeinen nicht output-sensitiv in Polynomialzeit durchgeführt werden können, außer $P = NP$. Dies sind zwar Negativresultate aber positiv betrachtet zeigen sie wie wichtig es ist, die output-sensitive Komplexität von Eckenauszähung zu verstehen, weil viele grundlegende und in der Praxis wichtige Probleme nur effizient gelöst werden können, indem in einem Zwischenschritt die Darstellung des Eingabepolytops in eine bestimme Form konvertiert wird.

Viele Ergebnisse in dieser Arbeit beantworten von Anderen aufgeworfene Fragen. In Kapitel 7 leiten wir viele neue Härteresultate her, die mit der Komplexität des Problems zu prüfen, ob eine gegebene Menge von polyhedrischen Kegel eine gegebene Menge[1] D überdeckt, zu tun haben. Als Korollar beantworten wir eine Frage von Bemporad et. al. [BFT01] negativ, indem wir beweisen, dass es NP-schwer ist zu entscheiden, ob die Vereinigung einer gegebenen Menge von $\mathcal{H}$- oder $\mathcal{V}$-Polytopen konvex ist. In ähnlicher Weise beantworten wir in Kapitel 8 eine Frage von Kaibel und Swartz [KS03], indem wir zeigen, dass es Graphisomorphie-vollständig ist zu prüfen, ob ein durch seine Facetten und Ecken gegebener Polytop kombinatorisch isomorph zu seinem polaren Dual ist.

Eine weiterer neuer Beitrag in Kapitel 8 ist die Konstruktion einer Familie von selbst-dualen Polytopen mittels free-joins. Die konstruierten Polytope sind für sich schon interessant, weil sie symmetrisierbare Inzidenzmatrizen haben. Außerdem geben wir ein Beispiel einer Klasse von Polytopen an, die wir die "Dach-Prismen" (roofed-prisms) nennen, die selbst-dual sind, aber sich nicht mittels freiem Zusammenfügen aus anderen Polytopen ergeben. Die Beobachtung, dass solche Polytope existieren, ist möglicherweise nicht überraschend, aber Dach-Prismen bilden eine interessante Klasse solcher Polytope.

VE-Vollständigkeit

In dieser Arbeit haben wir eine Komplexitätsklasse für Probleme definiert, die äquivalent zum Problem der Eckenauszähung sind. Diese Klasse versucht den Begriff der Vollständigkeit für VE in gleicher Weise zu fassen, wie die Klasse der NP-vollständigen Probleme es für die Probleme in NP tut. In Kapitel 6 wird definiert, was es heißt, dass ein Problem äquivalent zu VE bzw. schwerer oder leichter als VE ist. Eine Klasse von Problemen, die verwandt zur Berechnung der Projektion von Polytopen sind, wird als Beispiel für diese Begriffe verwendet. Es wird gezeigt, wie die verschiedenen Versionen des Projektionsproblems sich als NP-schwer, VE-vollständig, VE-schwer oder VE-leicht herausstellen.

Trotz mehrjähriger intensiver Forschung ist weder bekannt, ob VE zu P gehört, noch ob das Problem NP-schwer ist. Aufgrund dieser Tatsache glauben wir, das es sich als für die zukünftige Forschung nützlich herausstellen wird, einen solchen Begriff ist definieren. Diese Arbeit stellt zwei Probleme vor, die sich als verwandt zu dieser neuen Komplexitätsklasse herausstellen (Kapitel 6, Kapitel 7) und wir hoffen, dass für viele Probleme gezeigt wird, dass ihre Komplexität zu der des Problems der Eckenauszähung verwandt ist. Solche neuen Probleme werden es hoffentlich ermöglichen, Werkzeuge aus verschiedenen Gebieten zu nutzen, um die Frage der Komplexität der Eckenauszähung zu klären.

[1]Die Mengen, die wir explizit betrachten, sind entweder der gesamte Raum oder ein linearer Unterraum, obwohl der Fall, dass D ein Polytop ist, mit leichten Modifikationen behandelt werden kann.

Verhältnis VE zu GI

Es ist durchaus möglich, dass die Komplexität von Eckenauszählung irgendwo zwischen P und NP-vollständig liegt. Wir kennen keine Arbeit, die die Komplexität von Eckenauszählung zur Komplexität eines Problems wie Graphisomorphie in Beziehung zu setzen versucht. Ganz wie Eckenauszählung hat auch Graphisomorphie eine lange Forschungsgeschichte und seine Komplexität ist unbekannt. Einige Ergebnisse dieser Arbeit versuchen einen Schritt dahin zu gehen, eine Verbindung zwischen diesen beiden Problemen herzustellen. Auch wenn wir die Frage nicht abschließend beantworten, ob diese Probleme ähnliche Komplexität haben, glauben wir doch, dass die Ergebnisse in Kapitel 8 ein Schritt in die richtige Richtung sind.

Insbesondere zeigen wir in Kapitel 8, dass ein bestimmtes Isomorphieproblem für Polytope genau dann Graphisomorphie-vollständig ist, wenn Eckenauszählung leicht für Graphisomorphie ist. Für das Problem, das wir SD oder Selbstdualität nennen, wird gezeigt, dass es schwer für Eckenauszählung und auch für Graphisomorphie ist. Im Gegensatz zu anderen in dieser Arbeit betrachteten Problemen, von denen gezeigt wird, dass sie NP-schwer sind, deren NP-Schwere aber nicht zum Verständnis der Komplexität von Eckenauszählung beiträgt, wird das Problem der Selbstdualität nichttriviale Einsichten über die Komplexität von Eckenauszählung gewähren, je nachdem es sich als vollständig für Graphisomorphie oder echt schwerer als Graphisomorphie (z.B. NP-schwer) herausstellt.

Thesis Summary

Any polytope $P \subset \mathbb{R}^d$ can be represented as either the convex hull of a finite number of points in $\mathbb{R}^d$ or as a bounded intersection of a finite number of halfspaces. Either of these representations defines the other uniquely if redundancies are not allowed, but no output-sensitive algorithms are known that convert one of these representations to the other. Also, depending on whether the vertex representation or the facet representation of a polytope is known, the computational complexity of an operation can vary wildly. This thesis considers various computational problems about polytopes and their complexity.

The problems that we consider in this thesis, and the results that we obtain, have the following common themes:

- Many of the problems considered in this thesis are very fundamental operations on polytopes. Operations like the intersection, Minkowski addition and projection are very basic operations on polytopes that need to be frequently computed in practice. The complexity results in this thesis show that often the best way to perform these operations is by converting the representation of the input polytope first, highlighting the importance of finding an output-sensitive vertex enumeration algorithm.

- Every problem considered in this thesis is closely related to the vertex enumeration problem. For most problems, an efficient algorithm would imply and output-sensitive algorithm for vertex enumeration. Unfortunately, most of these problems turn out to be NP-hard or $\#P$-hard. The problem of checking whether a polytope is isomorphic to to its polar dual (chapter 8), on the other hand, relates the vertex enumeration problem with the well know graph isomorphism problem. Admittedly though the connection established in this thesis is somewhat weak.

- Every problem highlights the impact of input representation for computational problems about polytopes. For example, computing the vertex centroid (chapter 4) for a $\mathcal{V}$-polytope is trivial but is $\#P$-hard for $\mathcal{H}$-polytopes. Similarly, computing the facets of the intersection of two $\mathcal{H}$-polytopes is easy but the problem is NP-hard for almost all other representations.

The main contributions of this thesis are roughly divided into different categories as mentioned in the following subsections.

Strengthening previous results

In chapter 3 we show that enumerating the vertices of an unbounded $\mathcal{H}$-polyhedron P is NP-hard even when P has only 0/1 vertices. This strengthens a previous re-

sult of Khachiyan et. al. [KBB$^+$06] for general polyhedra. For general $\mathcal{H}$-polytopes the complexity of enumerating all the vertices is unknown but for a polytope all whose vertices are 0/1 there exists an output-sensitive algorithm [BL98]. Our result, thus, provides a better contrast between the complexities of vertex enumeration problem for polytopes and unbounded polyhedra.

Our small but very crucial modifications of the proof in [KBB$^+$06] allows us to obtain various other hardness results. For example, among other things, our method shows that it is not possible to identify in polynomial time whether an $\mathcal{H}$-polyhedron is a 0/1-polyhedron or not unless $P = NP$. Many of these additional results in chapter 3 are already known but our method allows us to unify all these results, and although these additional results in themselves are not an improvement over existing results, the unifying proof can arguably be considered an improvement.

In chapter 8 we prove that it is graph isomorphism complete to check if a polytope, given by $\mathcal{HV}$-representation, is combinatorially isomorphic to its polar dual. This generalizes a previous result of Kaibel and Swartz [KS03] proving the GI-completeness of checking combinatorial isomorphism of two $\mathcal{HV}$-polytopes. Our result is based on a simple observation characterizing the decomposability of a polytope as free-join of smaller polytopes (Lemma 8.5.1).

New Hardness results

The most important new results in this thesis concern proving that many basic operations on polytopes like computing intersections, Minkowski sums, projection etc can not in general be performed in output-sensitive polynomial time unless $P = NP$. Although these results are negative, they do provide a positive view for algorithmic research on polytopes - it is very important to understand the output-sensitive complexity of vertex enumeration because many basic and practically important problems can only be efficiently solved via an intermediate step of converting the representation of the input polytopes to a certain form.

Many results in this thesis answer questions raised by others. In chapter 7 we derive many new hardness results related to the complexity of checking whether a given set of polyhedral cones cover a given set[2] D. As a corollary we answer a question of Bemporad et. al. [BFT01] in the negative by proving that it is NP-hard to decide whether the union of a given set of $\mathcal{H}$- or $\mathcal{V}$-polytopes is convex. Similarly, in chapter 8 we answer a question of Kaibel and Swartz [KS03] by proving that it is graph isomorphism complete to check whether a polytope, given by its

[2]The sets that we explicitly consider are either the whole space or a linear subspace, although one can handle the case when D is a polytope with slight modifications.

facets and vertices, is combinatorially isomorphic to its polar dual.

Another novel contribution in chapter 8 is the construction of a family of self-dual polytopes via free-join. The polytopes constructed are interesting on their own because their incidence matrices are symmetrizable. In general transposability of the incidence matrix suffices for self-duality of a polytope and not all self-dual polytopes have symmetrizable incidence matrices. We also provide an example of a class of polytopes, which we call the roofed-prisms, that are self-dual but do not arise as a free-join of other polytopes. The observation that such polytopes exist is perhaps not surprising but roofed-prisms form an interesting class of such polytopes.

Completeness for VE

In this thesis we have defined a complexity class for problems that are equivalent to the vertex enumeration problem. This class tries to capture the notion of completeness for VE in the same way the class of NP-complete problems does for problems in NP. In chapter 6 the notion of a problem being equivalent, harder or easier than VE is defined, and a class of problems related to computing the projection of polytopes is used as an example for these notions. It is shown how the various versions of the projection problem turn out to be either NP-hard, VE-complete, VE-hard or VE-easy.

Despite years of active research neither it is known whether VE is in P nor it is known to be NP-hard. This fact makes us believe that defining such a notion of completeness will turn out to be useful for future research. This thesis presents two problems that turn out to be related to this new complexity class (chapter 6, chapter 7) and we hope that many problems will be shown to have a complexity related to that of the vertex enumeration problem. Hopefully such new problems will make it possible to use tools from different areas for settling the question of the complexity of vertex enumeration.

Relating VE to GI

It is entirely possible that the complexity of vertex enumeration lies somewhere between P and NP-complete. We are not aware of any work trying to relate the complexity of vertex enumeration to some problem like graph isomorphism. Graph isomorphism, like vertex enumeration, has a long history of research and an unknown complexity status. Some results in this thesis try to take a step towards establishing the connection between these two problems. Even though we do not settle the question whether these two problems have similar complexities, we believe the results in chapter 8 are a step in the right direction.

In particular, we show in chapter 8 that a certain isomorphism problem defined for polytopes is graph isomorphism complete if and only if vertex enumeration is graph isomorphism easy. The problem, which we call SD or Self Duality, is shown to be vertex enumeration hard as well as graph isomorphism hard. As opposed to other problems considered in this thesis, that are shown to be NP-hard but whose NP-hardness offers no insight into the complexity of vertex enumeration, the self-duality problem will provide non-trivial insight into the complexity of vertex enumeration whether it turns out to be graph isomorphism complete or strictly harder that graph isomorphism (say NP-hard).

Contents

Chapter 1

Introduction

A *convex polyhedron* is an intersection of a finite number of halfspaces $\mathcal{H}$. A polyhedron is called *pointed* if it does not contain an affine line and *bounded* if it does not contain a ray. The Minkowski-Weyl theorem for polyhedra states that every polyhedron can be represented as $conv(V) + cone(Y)$, where V and Y are finite sets of vectors in $\mathbb{R}^d$, and $conv(X)$ and $cone(X)$ are respectively the convex and the conic hull of a set X. Bounded polyhedra are called polytopes, and for polytopes we have $Y = \emptyset$. Although a polyhedron can be bounded or unbounded, in this thesis we use the word polyhedron exclusively for the unbounded case; for the bounded case we use the term polytope.

The representation of a polyhedron as intersection of halfspaces is called the $\mathcal{H}$-representation and the representation as the convex hull of points is called the $\mathcal{V}$-representations. An $\mathcal{H}$-representation is called *minimal* if omitting some halfspace from $\mathcal{H}$ does not change the polyhedron. Similarly a V-representation is called minimal if omitting some vector in the sets V or Y does not change the polyhedron. For any pointed polyhedron the minimal $\mathcal{H}$- or $\mathcal{V}$-representations are unique and the $\mathcal{H}$-representation completely defines the $\mathcal{V}$-representation, and vice-versa.

Although the $\mathcal{H}$- and the $\mathcal{V}$-representations are equivalent, that is they define the same convex set and one completely determines the other, they are quite different from a computational perspective. Consider the following problem for a polytope P: Given a vector $c \in \mathbb{R}^d$, find a point $x \in P$ such that the objective value $c^T x$ is maximized, i.e., we are interested in a point $x \in P$ such that $c^T x \geq c^T y$, for all $y \in P$. It is well known that the point x^* achieving the maximum can always be assumed to be one of the vertices of P and thus given the $\mathcal{V}$-representation of

P, this problem can be trivially solved by evaluating $c^T v$ for each vector $v \in V$ and taking the maximum. But if P is given in $\mathcal{H}$-representation the problems becomes fairly non-trivial. In fact, the problem for $\mathcal{H}$-representation is known as the Linear Programming problem which has been a subject of extensive research since the time of Fourier.

Perhaps the previous example does not fully convey how different the computational complexity of the same operation can be for these two representations. From the example of linear programming, one might be tempted to think that it is always a case of triviality versus existence of a fairly non-trivial yet polynomial algorithm. After all, even though a polynomial algorithm for linear programming was found as recently as 1979 [Kha79] and involved using powerful techniques developed over a long period, we nevertheless have a polynomial algorithm. This is not true in general and as we will see in this thesis, many computations on polytopes are in fact NP-hard (or even #P-hard) in one representation and polynomially solvable in the other. Note that this disparity in the complexity occurs because the number of vertices and the number of facets of a polytope can vary greatly. Indeed as we will see in thesis, if the dimension is not assumed to be a fixed constant then one representation can be exponential compared to the other representation.

Since the representation of a polytope plays a crucial role in how efficiently one can perform a computation on a given polytope, it seems natural from a computational perspective to find an algorithm that converts one representation to the other efficiently. The problem of representation conversion is fundamental from a theoretical viewpoint as well - as highlighted by the observation that proving the equivalence of the two representations of a polytope, generally involves giving a procedure to convert one representation to the other and then proving termination of this procedure. The problem of converting the $\mathcal{H}$-representation to the $\mathcal{V}$-representation is known as the Vertex Enumeration problem (VE) and the problem of converting the $\mathcal{V}$-representation to the $\mathcal{H}$-representation is called the Convex Hull problem (CH). The two problems are equivalent and an instance of one can be converted to an instance of another if a point in the interior of the polytope is known. Finding such a point for the $\mathcal{V}$-representation is trivial while for $\mathcal{H}$-representation one can find such a point via linear programming.

It is known that a d-dimensional polytope with n halfspaces (vertices resp.) in its minimal representation can have as many as $\binom{n-\lfloor \frac{d+1}{2} \rfloor}{n-d} + \binom{n-\lfloor \frac{d+2}{2} \rfloor}{n-d}$ vertices (facets resp.) [McM70]. The dominating term in the upper bound theorem is exponential in d and thus one can not hope to find a polynomial algorithm for this problem if the dimension is not fixed. Also, since the upper bound theorem is symmetric in the number of vertices and facets, we have a tremendous gap between the maximum and the minimum number of vertices of a polytope in terms of the number of facets and the dimension. This fact suggests that one needs to measure the running time of any algorithm for the problem of representation conversion both in terms of the input and the output sizes. An algorithm whose computational complexity is a function of both the input and the output sizes is called an *output sensitive* algorithm and this notion of complexity of an algorithm is frequently used for enumeration problems. Naturally, whenever possible one is interested in finding an algorithm whose complexity is only a polynomial in the input and the output sizes. In this thesis any discussion about output sensitivity assumes that we are interested in polynomial complexity and thus we omit the word polynomial and use output-sensitive to mean "polynomial in the size of the input and the output".

Various algorithms are known for the representation conversion problem but none of them are output sensitive for general polytopes [ABS97]. This thesis started out with the goal of finding an output sensitive algorithm for Vertex Enumeration problem but unfortunately such an algorithm remains elusive. In the process of trying to find such an algorithm I discovered many ways that do not give an output sensitive algorithm and this thesis shaped up to become a collection of negative results about various computational tasks relating to polytopes. A typical chapter in this thesis starts with the discussion of a particular problem on polytopes or polyhedra that relates in some way to the problem of vertex enumeration. We discuss this connection briefly and then proceed to study that particular problem. Since most of our results are about hardness of such problems, and thus do not yield a solution to vertex enumeration, we discuss the problems and associated results in a way somewhat independent of vertex enumeration.

In Chapter 3 we consider the problem of enumerating the vertices of a polyhedron. This is harder than computing both the vertices V and the extreme rays Y of a polyhedron. Khachiyan et. al. [KBB$^+$06] proved this problem to be NP-hard. We strengthen this result and prove that this problem remains NP-hard for the special class of polyhedra all whose vertices are a subset of the vertices of a hypercube. The vertices of such a polyhedra can be represented only using

0 and 1 as coordinate values and such polyhedra are aptly named 0/1-polyhedra. Our result can be contrasted to the results of Bussieck and Lübbecke [BL98] who showed that for 0/1-polytopes the vertices can be enumerated in polynomial time. As mentioned before, the problems of enumerating vertices of a general polytope remains open.

We build upon the proof in [KBB$^+$06] and by analyzing a polyhedron associated with the negative-flows in a directed graph we are able to derive many other hardness results. Apart from strengthening the results of [KBB$^+$06], we show that checking whether an $\mathcal{H}$-polyhedron is 0/1 or not is NP-hard. Ding, Feng and Zang [DFZ07] have already shown that checking whether an $\mathcal{H}$-polyhedra is 0/1 or not is NP-hard, but their construction results in polyhedra with exponentially many vertices. The polyhedra arising in our construction, on the other hand, have only a polynomial number of vertices. We also show that checking half-integrality of an $\mathcal{H}$-polyhedron is NP-hard as well. A polyhedron $P \subseteq \mathbb{R}^n$ is said to be $\frac{1}{f}$-*integral* [Vaz01] if $\mathcal{V}(P) \subseteq \{0, \frac{1}{f}, \frac{1}{f-1}, \ldots, \frac{1}{2}, 1\}^n$. In particular, for $f = 2$, such a polyhedron is called *half-integral*.

The same construction also shows that the maximum support of a d-dimensional polyhedron can not be approximated to any factor less than $\frac{d}{12}$. For a polyhedron $P = \{x \in \mathbb{R}^d : \ Ax = b, \ x \geq 0\}$, the support of a vertex $v \in \mathcal{V}(P)$ is defined as the number of positive components of v. Finding a vertex of a polyhedron with maximum support includes several interesting problems, such as MAX-CUT in undirected graphs, and LONGEST-CYCLE in directed graphs.

In Chapter 4 we consider the problem of computing the vertex centroid of a polytope. The vertex centroid is defined as the average of the vertices of a polytope, i.e. if $c(P)$ be the centroid of the polytope P then

$$c(P) = \sum_{v \in \mathcal{V}(P)} \frac{v}{|\mathcal{V}(P)|}.$$

The relationship of the vertex centroid with the number of vertices bears a parallel to the relation of the center of mass and the volume of a polytope. Computing the center of mass was recently shown to be $\#P$-hard by Rademacher [Rad07], even though it can be approximated quite well via random sampling [KLS98]. Similarly, computing the volume of a polytope is known to be $\#P$-hard [DF88] even though the volume can be approximated well by random sampling [KLS98]. Computing the number of vertices of a polytope is also known to be $\#P$-hard

[Lin86] but approximating it is an open problem. We show that computing the vertex centroid is also $\#P$-hard and approximating it is $\#P$-easy. For polyhedra we further show that approximating the vertex centroid to any sufficiently non-trivial degree is hard.

In Chapter 5 we consider the complexity of computing the Minkowski sum, the intersection, or the convex hull of the union of two polytopes. The Minkowski sum $P_1 + P_2$ of two polytopes P_1 and P_2 is defined as follows:

$$P_1 + P_2 = \{x + y | x \in P_1, y \in P_2\}.$$

Our motivation for considering these operations arise from the fact that vertex enumeration can be solved in polynomial time if these operations can be performed efficiently. Naturally many versions of these problems have trivial solution. For example, computing the vertices of the Minkowski sum of two $\mathcal{V}$-polytopes can be done easily via linear programming. We are interested in other variants of these operations where such easy solutions do not exist.

We consider these problems for various other input and output representations and derive various hardness results. For the Minkowski sum, we prove that enumerating the facets of $P_1 + P_2$ is NP-hard if P_1 and P_2 are $\mathcal{H}$-polytopes, or if P_1 is specified by vertices and P_2 is a polyhedral cone specified by facets. For the intersection, we prove that computing the facets or the vertices of the intersection of two polytopes is NP-hard if one of them is given by vertices and the other by facets. Also, computing the vertices of the intersection of two polytopes given by vertices is shown to be NP-hard. Analogous results for computing the convex hull of the union of two polytopes follow from polar duality.

Another fundamental operation on polytopes is the projection of a d-dimensional polytope onto an affine subspace of dimension $k \leq d$ for an arbitrary k. In Chapter 6 we consider the computational complexity of this operation. We show that computing either the vertices or the facets of the projection of an $\mathcal{H}$-polytope is NP-hard while computing both the vertices and the facets of the projection is equivalent to the vertex enumeration problem. The NP-hardness results in this chapter follow from the results in Chapter 5, nevertheless the discussion about projection is included into a separate chapter due to the fact that projection in itself is a fundamental operation in the theory of polytopes and we consider it worthwhile to include a coherent and complete discussion of this operation. We also discuss the case when the given subspace is not arbitrary but picked at random. In contrast to the case of projection onto arbitrary subspaces and somewhat surprisingly, computing the facets of the projection of a polytope onto a randomly picked subspace can be done in polynomial time even though computing the vertices remains vertex enumeration hard. The reader should note that we are not necessarily interested in a randomized algorithm but only the complexity of computing the projection for instances that picked with a certain random distribution.

The projection operation also serves as an introductory example for the notion of VE-complete, -hard, or -easy problems which we introduce in Chapter 6. The idea of defining such classes is to identify the complexity of problems relative to the vertex enumeration problem. Thus, for example, a VE-complete problem is equivalent to vertex enumeration and a polynomial algorithm for the vertex enumeration problem would give a polynomial algorithm for the VE-complete problem at hand, and vice-versa. The definition of such classes is motivated again by the fact that despite years of active research the complexity status of the vertex enumeration problem remains unknown, and it might be fruitful to try to relate this problem with other problems.

In Chapter 7, we consider another problem whose complexity is closely related to the vertex enumeration problem. The problem, which we call the Cone-Cover problem, asks us to decide whether a given set of polyhedral cones, each embedded in $\mathbb{R}^d$, covers a given convex set D. We discuss the cases where D is either the whole space $\mathbb{R}^d$ or an affine subspace $\mathbb{R}^k$. It turns out that the complexity of this problem depends on how the cones intersect with each other. We consider various cases and prove NP-hardness of some cases and provide polynomial algorithms for some other cases.

The motivation for considering this problem is twofolds. Firstly, some variants of this problem have very close relation with the vertex enumeration problem and they provide another example for the VE-hard class defined in chapter 6. Secondly, as a corollary of our results we obtain that checking whether the union of a given set of polytopes is convex or not, is hard if the polytopes are given only by their vertices or their facets. This answers in negative a question of Bemporad, Fukuda and Torrisi [BFT01] about whether or not a polynomial algorithm exists for checking if the union of a finite number of polytopes in $\mathbb{R}^d$ is convex.

Finally, in Chapter 8 we try to relate the complexity of vertex enumeration to that of graph isomorphism. This is motivated by the fact that, as we will see in this thesis, problems that are related to vertex enumeration frequently turn out to be NP-hard but at the same time the NP-hardness of such problems offers no useful insight into the complexity of the vertex enumeration problem. To avoid this we try to relate the complexity of vertex enumeration with that of graph isomorphism. As it turns out, although we can not yet say much about vertex enumeration, there is a natural computational question about polytope isomorphism that relates the complexity of the vertex enumeration problem and the graph isomorphism problem.

In particular, we consider the problem of checking whether a polytope is self-dual or not. A polytopes is called self-dual if its face-lattice is ismorphic to that of its polar dual. Kaibel and Schwartz [KS03] showed that given two polytopes by their $\mathcal{H}$- as well as $\mathcal{V}$-representation, it is graph isomorphism complete to decide whether these two polytopes are combinatorially isomorphic to each other, i.e. whether their face lattices are isomorphic or not. We extend their result and show that two polytopes P, Q are ismorphic if and only if the free-join of P with the polar of Q is self-dual. This proves that checking self-duality of a polytope given by both vertices and facets is graph isomorphism complete. As a corollary we obtain that checking self-duality of a polytope given by only $\mathcal{H}$- or $\mathcal{V}$-representation is graph isomorphism complete if and only if vertex enumeration is graph isomorphism easy. The polytopes arising in our construction are not only self-dual but also have the property that their facet-vertex incidence matrices are symmetrizable. For self-duality transposability of the incidence matrix suffices and it is known that not all transposable matrices are symmetrizable [BGZ06]. We also present a class of polytopes that are self-dual but whose incidence matrix is not symmetrizable.

Chapter 2

Preliminaries

Given a set $X \subset \mathbb{R}^d$, the affine, the conic and the convex hull of X are defined as

$$\begin{aligned} \mathit{aff}(X) &= \left\{ x \in \mathbb{R}^d \,\middle|\, \sum_{\alpha \in X} \lambda_\alpha \alpha = x, \sum_{\alpha \in X} \lambda_\alpha = 1 \right\} && (2.1) \\ \mathit{cone}(X) &= \left\{ x \in \mathbb{R}^d \,\middle|\, \sum_{\alpha \in X} \lambda_\alpha \alpha = x, \lambda_\alpha \geq 0 \right\} && (2.2) \\ \mathit{conv}(X) &= \left\{ x \in \mathbb{R}^d \,\middle|\, \sum_{\alpha \in X} \lambda_\alpha \alpha = x, \sum_{\alpha \in X} \lambda_\alpha = 1, \lambda_\alpha \geq 0 \right\} && (2.3) \end{aligned}$$

A set X is called *bounded* if it contains no ray and *full dimensional* if $\mathit{aff}(X) = \mathbb{R}^d$.

2.1 Polytopes and Polyhedra

A polytope is the convex hull of a finite number of points. A basic result in polytope theory states that the set of polytopes is exactly the set of bounded intersections of a finite number of halfspaces ([Zie95], Theorem 1.1).

A polyhedral cone is the conic hull of a finite number of points. A polyhedral cone can also be represented as the intersection of a finite number of halfspaces each containing origin on its boundary.

A polyhedron $P \subset \mathbb{R}^d$ is defined as the intersection of a finite number of halfspaces.

$$P = \left\{ x \in \mathbb{R}^d \,|\, Ax \leq b \right\},$$

where $A \in \mathbb{R}^{m \times d}$ is an $m \times d$ matrix of real numbers and $b \in \mathbb{R}^{m \times 1}$ is a column vector. If the origin lies in the interior of P then each entry in b can be assumed to be 1. A polyhedron is called *pointed* if it does not contain any line. Note that a polyhedron need not be bounded.

The Minkowski-Weyl theorem [Sch86] for polyhedra states that every polyhedron can be represented as the Minkowski sum $conv(V) + cone(Y)$ of a polytope $conv(V)$ and a polyhedral cone $cone(Y)$. For polytopes we have $Y = \emptyset$ while for polyhedral cones the set V contains a single point.

A polyhedron (polytope) is called an $\mathcal{H}$-polyhedron ($\mathcal{H}$-polytope) if it is described as the intersection of halfspaces, and it is called a $\mathcal{V}$-polyhedron ($\mathcal{V}$-polytope) if it is described by the sets V and Y. Accordingly, these two representations are called the $\mathcal{H}$- and the $\mathcal{V}$-representation, respectively.

An $\mathcal{H}$-representation of a polyhedron is called non-redundant if omitting any halfspace does not change the polyhedron. Similarly the $\mathcal{V}$-representation is called non-redundant if omitting any vectors in V or Y does not change the polyhedron. For a pointed and full dimensional polyhedron, and hence for full dimensional polytopes and polyhedral cones, the non-redundant $\mathcal{H}$- and $\mathcal{V}$-representations are unique.

2.1.1 Faces and the face lattice of a polytope

For $\alpha \in \mathbb{R}^d, \beta \in \mathbb{R}$, the inequality $\alpha^T x \leq \beta$ is said to be a valid inequality for a polytope P if $P \subset \{x \in \mathbb{R}^d | \alpha^T x \leq \beta\}$. A set $F \subseteq P$ is called a face of P if there exists a valid inequality $\alpha^T x \leq \beta$ such that $F = \{x \in P | \alpha^T x = \beta\}$. In this case we say that F is the face defined (or induced) by $\alpha^T x \leq \beta$. The 0-dimensional faces are called the vertices, 1-dimensional faces the edges, $(d-2)$-dimensional faces the ridges and the $(d-1)$-dimensional faces are called facets. For the inequalities $\mathbb{O}^T x \leq 0$ and $\mathbb{O}^T x \leq 1$ we obtain that P itself and the empty set $\emptyset$ are both faces of the polytope. A face f of P is called a proper face if it is neither the empty set nor P itself.

A hyperplane $h = \{x \in \mathbb{R}^d | \alpha^T x = \beta\}$ is called a supporting hyperplane of a polytope P if the inequality $\alpha^x \leq \beta$ is not only valid for P but also induces a proper face. A supporting hyperplane h is called a *facet defining* hyperplane of P if $P \cap h$ is a facet of P. For every facet of P there is a unique facet defining hyperplane. The vertices of P are denoted by $\mathcal{V}(P)$ and the facet defining hyperplanes are denoted by $\mathcal{H}(P)$. We will usually not differentiate between a facet of P, its defining hyperplane $\{x|\alpha^T x = \beta\}$, the inequality $\alpha^T x \leq \beta$ and the halfspace defined by this inequality, and $\mathcal{H}(P)$ will refer to any of these depending on the context. Since any of these interpretations unambiguously determines the others, quite often we will also switch from one interpretation of $\mathcal{H}(P)$ to another interpretation for simplicity of notation.

Let $\mathcal{F}(P)$ denote the set of faces of a polytope P. The poset $(\mathcal{F}(P), \subset)$ defines a lattice and is called the face lattice of P. This lattice, denoted by $L(P)$, can be visualized by the Hasse diagram where the faces of polytope are arranged in levels with P at the top (level 0). A level at depth i contains all the faces of dimension $d - i$ and thus the empty set is the $(d + 1)$-st level. An i-level face F has an edge with an $(i + 1)$-level face G if and only if $F \supset G$. An example the face lattice of a square pyramid is shown in Figure 2.1.

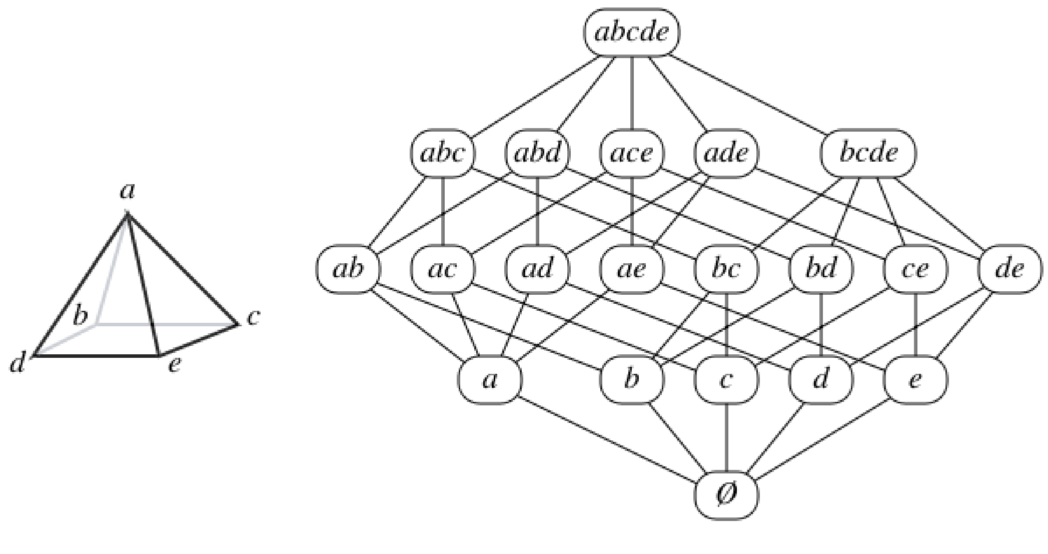


Figure 2.1: Face lattice of a square pyramid

Another equivalent characterization of the vertices of a polytope is the following. A *vertex* or an *extreme point* of a polytope P is a point $v \in \mathbb{R}^d$ that cannot be represented as a convex combination of two other points of P, i.e., there exists no $\lambda \in (0, 1)$ and $v_1, v_2 \in P$ such that $v = \lambda v_1 + (1 - \lambda)v_2$. For a polyhedron P, a *direction* of P is a vector $r \in \mathbb{R}^d$ such that $x_0 + \mu r \in P$ whenever $x_0 \in P$ and $\mu \geq 0$. An *extreme direction* of P is a direction r that cannot be written as a conic combination of two other directions, i.e., there exist no non-negative numbers $\mu_1, \mu_2 \in \mathbb{R}_+$ and directions r_1, r_2 of P such that $r = \mu_1 r_1 + \mu_2 r_2$. We denote the sets of extreme points and directions of a polyhedron P by $\mathcal{V}(P)$ and $\mathcal{D}(P)$ respectively.

2.2 Polarity

For the discussion in this thesis, it is convenient to assume that the origin is contained in the given polyhedron and hence the $\mathcal{H}$-representation of a polyhedron $P \subset \mathbb{R}^d$ can always be assumed to be of the form

$$\begin{aligned} Ax &\leq \mathbf{1} \\ Bx &\leq \mathbf{0} \end{aligned}$$

The polar dual of P denoted by P^* is obtained by interpreting each row of the matrices A and B as points in $\mathbb{R}^d$ and is defined as the Minkowski sum

$$conv\left(A \cup \{\mathbb{O}\}\right) + cone\left(B\right).$$

Theorem 2.2.1. *Let $P \subset \mathbb{R}^d$ be a polyhedron with $\mathcal{H}(P) : Ax \leq \mathbf{1}, Bx \leq \mathbf{0}$ and $\mathcal{V}(P) : conv(V) + cone(Y)$ then, for the polar P^**

$$\begin{aligned} \mathcal{H}(P^*) &: Vx \leq \mathbf{1} \\ & Yx \leq \mathbf{0} \\ \mathcal{V}(P^*) &: conv(A \cup \{\mathbb{O}\}) + cone(B) \end{aligned}$$

Here we list some properties of this operation. We omit the proofs and direct the reader to [Zie95] for details.

Properties of the polar operation:

(i) $P \subseteq Q$ implies $P^* \supseteq Q^*$,

(ii) $P = P^{**}$,

(iii) $\mathbf{0} \in P^*$

Let $P = conv(V) = \{Ax \leq \mathbf{1}\}$ be a polytope in $\mathbb{R}^d$, and let

$$F = conv(V') = \{x \in \mathbb{R}^d | A'x = \mathbf{1}, Ax \leq \mathbf{1}\}$$

be a face of P, with $V' \subset V$ and $A' \subset A$. Define,

$$F^* = conv(A') = \{x \in \mathbb{R}^d | V'x = \mathbf{1}, Vx \leq \mathbf{1}\}.$$

Then, we have the following:

Theorem 2.2.2. *Let P be a polytope with $\mathbf{0} \in int(P)$, and let F, G be some faces of P. Then*

(i) F^* is a face of P^*,

(ii) $F^{**} = F$,

(iii) $F \subseteq G$ if and only if $F^* \supseteq G^*$.

Corollary 2.2.3. *The face lattice of P^* is the opposite of the face lattice of P.*

2.3 Model of Complexity

For the polyhedral computations that we consider in this thesis, the input polyhedron is described either by the set of its vertices and extreme rays, i.e. the sets V and Y, or the set of inequalities $Ax \leq \mathbf{1}, Bx \leq \mathbf{0}$, or both. We define the *size* of the polyhedron to include the number of entries in the matrices V, Y, A and B. Sometimes, though, we will include the number of bits, L, required to represent the largest entry in any of these matrices, in the notion of the size of the input. When we do not include the parameter L in our analysis, we will assume the real-RAM model, and when we include L we assume the bit-model. Both of these models are described briefly in the following subsections. A more precise treatment of the real RAM model is available in Preparata and Shamos [PS85]. The bit-model of computation is described by Braverman and Cook in [BC06].

2.3.1 Real RAM model

In the real RAM (Random Access Machine) model the abstract computational machine consists of a (potentially) infinite memory consisting of cells (or words) and any arbitrary memory cell can be accessed for reading or writing in constant time. Each memory cell can hold a real number and the machine can perform some "basic" arithmetic operations, like addition or multiplication, on two words in constant time.

Thus, in such a model, the input consists of a sequence of real numbers and the complexity (or time complexity) of an algorithm is measured in terms of the number of basic arithmetic operations performed by the algorithm. Naturally the power of such a machine also depends on the set of the basic operations that the machine is equipped with. In the text "Computational Geometry: An Introduction" [PS85], Preparata and Shamos list the following operations as the basic operations:

1. The arithmetic operations $(+, -, \times, /)$.
2. Comparisons between two real numbers $(<, \leq, =, \neq, \geq, >)$.
3. Indirect addressing of memory (integer addresses only).
4. k-th root, trigonometric functions, EXP, and LOG (in general, analytic functions).

2.3.2 Bit-model

Since the assumption that one word of the real RAM machine can hold any arbitrary real number might be unrealistic in certain scenarios, sometimes we will consider the bit-model where each real number in a word is approximated by a rational number which in turn can be represented by a sequence of bits encoding the numerator and the denominator in some way. The size of a number denotes the number of bits required to represent the numerator and the denominator, and the time to perform any basic operation on two numbers is specified as a function of the size of the operands. In essence, the bit-model requires that while working in the real-RAM model we can only read any real number x in form of a rational number $\frac{p}{q}$ that approximates x to some given precision, and that we write back rational approximations of the real numbers to the memory.

The bit-model becomes necessary in the following scenario: Suppose one wants to solve problem Φ_1 by reducing it to another problem Φ_2. Since for realistic computers one has to spend time proportional to the binary encoding for processing and operating on a number, one has to argue that the actual numbers involved in the representation of Φ_2 are not much larger than those in the representation of Φ_1. An example of this can be seen in chapter 5.

2.3.3 Output-sensitivity

One crucial difference between complexity notions in this thesis and the standard notions is that in this thesis the time complexity will generally be not treated as a function of the input size only but also of the output size. For decision problems the output is a single bit, while for enumeration problems the output size can have a wide range. For example, for a given set S of points on the plane the vertices of their convex hull could either be just three points or all of the input points. If n denotes the number of input points and h denotes the number of vertices of the convex hull of S, then an $O(n \log h)$ algorithm is arguably better than an $O(n \log n$ algorithm even though in the worst case they have similar complexity.

For the vertex enumeration problem the effect of considering output-sensitivity is more pronounced. Roughly speaking a d-dimensional polytope defined by n inequalities can have as many as $O(n^{\lfloor \frac{d}{2} \rfloor})$ vertices but as few as $O(n^{\frac{1}{\lfloor \frac{d}{2} \rfloor}})$ vertices where the big-oh notation hides polynomials in d. Even though worst case optimal algorithms with time complexity $O(n^{\lfloor \frac{d}{2} \rfloor})$ exist [Cha93], if the dimension is not assumed to be a fixed constant then the worst case optimal algorithm is exponential in d even though the output size need not always match this worst case bound. Indeed, for all existing vertex enumeration algorithms there are classes of polytopes such that the algorithm takes time that is not polymial in the sizes of the input and the output for variable dimension [ABS97]. Clearly an output sensitive algorithm with polynomial dependency on the input and the output can be of great utility for cases where the behavior is far from the worst case.

Chapter 3

Vertices of Polyhedra

In this chapter we consider the problem of enumerating all vertices of an unbounded polyhedron. This is a joint work with Endre Boros, Khaled Elbassioni and Vladimir Gurvich, and is an extension of the previous work of Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni and Vladimir Gurvich [KBB$^+$06]. The results in this chapter have been accepted for publication in the journal Annals of Operations Research and have appeared as Rutgers research report [BEGT08].

3.1 Introduction

It is known that the problem of enumerating the vertices of an $\mathcal{H}$-polytope is equivalent to the following decision problem: Given an $\mathcal{H}$-polytope P and a subset V of its vertices, is $V = \mathcal{V}(P)$?

Consider the following approach to solving this decision problem. Pick any arbitrary facet F of P and the subset $V' \subset V$ of vertices that lie on F. F itself is a polytope of dimension $dim(P) - 1$. Verify recursively whether $V' = \mathcal{V}(F)$ or not. We can decide efficiently if $V = \mathcal{V}(P)$ provide the following problem can be solved in polynomial time:

Given an $\mathcal{H}$-polytope P, a facet F of P and a subset $V \subseteq \mathcal{V}(P) \setminus \mathcal{V}(F)$, is $V = \mathcal{V}(P) \setminus \mathcal{V}(F)$?

Essentially this newer problem requires one to verify that a given subset of vertices of P includes all the vertices of P not lying on a particular facet F. We will now prove that this problem is equivalent to the problem of deciding whether a given subset of vertices of an unbounded $\mathcal{H}$-polyhedron includes all vertices of P.

Let the polytope P be described by the inequalities $Ax \leq \mathbf{1}$ and let the inequality corresponding to F be $a \cdot x \leq 1$. P can be transformed into a combinatorially isomorphic polytope P' such that removing the inequality for the facet F' corresponding to F creates an unbounded polyhedron. To see this, recall the following facts about polar duality:

Fact 1: Given a polyhedron $\{Ax \leq \mathbf{1}, Bx \leq 0\}$, the polar dual $conv(A \cup \{\mathbb{O}\}) + cone(\{B\})$ contains the origin.

Fact 2: For a polyhedron (or polytope) P, the polar dual is an unbounded polyhedron if and only if P does not contain the origin in its interior.

We move the origin to lie on F and only on F so as to obtain a polytope described by the inequalities $\{A'x \leq \mathbf{1}, a' \cdot x \leq 0\}$. The polar of this polytope is $conv(A' \cup \{\mathbb{O}\}) + cone(\{a'\})$. Move the origin again to ensure that it does not lie inside $conv(A')$. This gives another polyhedron $conv(A'') + cone(a'')$. Taking the polar once more we get a polytope P'' represented as $\{A''x \leq \mathbf{1}, a'' \cdot x \leq 0\}$ that is combinatorially isomorphic to P and removing the inequality corresponding to F (*i.e.* $\{a'' \cdot x \leq 0\}$) produces a polyhedron $\{A''x \leq \mathbf{1}\}$ that is unbounded since $conv(A'')$ does not contain origin.

Now consider the following problem:

Given an $\mathcal{H}$-polyhedron P and a subset $X \subseteq \mathcal{C}(P)$, is $X = \mathcal{C}(P)$?

In this formulation, $\mathcal{C}(P)$ could be either the set of vertices $\mathcal{V}(P)$, the set of extreme rays $\mathcal{D}(P)$, or the set of both the vertices and extreme rays $\mathcal{V}(P) \cup \mathcal{D}(P)$ of P. For the case when $\mathcal{C}(P)$ is $\mathcal{V}(P)$ this problem is equivalent to the earlier mentioned problem of verifying whether a given subset of vertices of a polytope include all the vertices of P except those that lie on a particular facet F. Furthermore, it is well-known and also easy to see that the decision problems for $\mathcal{D}(P)$ or for $\mathcal{V}(P) \cup \mathcal{D}(P)$ are equivalent to that for $\mathcal{V}(P')$ where P' is some polytope derived from P. It is also well-known that if the decision problem is NP-hard, then no output polynomial-time algorithm can generate the elements of the set $\mathcal{C}(P)$ unless P=NP (see e.g. [BEGM07]).

The complexity of some interesting restrictions of these problems have already been settled. Most notably, it was shown in [BL98], that for a 0/1-polytope, i.e., for which $\mathcal{V}(P) \subseteq \{0,1\}^n$, given in its $\mathcal{H}$-representation, the problem of finding the vertices given the facets can be solved with polynomial delay (i.e. the time to produce each new vertex is bounded by a polynomial in the input size) using a simple backtracking algorithm. Output-sensitive algorithms for vertex enumeration also exist for simple and simplicial polytopes [AF92, AF96, BFM98], for network polyhedra and their duals [Pro94], and for some other classes of polyhedra [ADP03].

More recently, it was shown in [KBB$^+$06] that the problem of generating the vertices of an unbounded polyhedron P is NP-hard in general. On the other hand, for special classes of 0/1-polyhedra, e.g. for the polyhedron of s-t-cuts in general graphs [GV95], for polyhedra associated with the incidence matrix of bipartite graphs, and for polyhedra associated with 0/1-network matrices [BEGM07], output-sensitive algorithms for the vertex enumeration problem can be obtained using problem-specific techniques.

This naturally raises the question whether there also exists an output-sensitive algorithm for enumerating the vertices of any 0/1-polyhedron, extending the result of [BL98] for 0/1-polytopes. Here we show that this is not possible unless $P = NP$. Our result strengthens the result of [KBB$^+$06], which did not apply to 0/1-polyhedra. It uses almost the same construction, but goes through the characterization of the vertices of the polyhedron of negative weight-flows of a graph, defined in the next section. We also characterize the extreme rays of this polyhedron and show that this polyhedron can have many more extreme rays compared to the number of its vertices.

Ding, Feng and Zang [DFZ07] have recently shown that distinguishing whether a polyhedron, given by its facets, is either a 0/1 polyhedron or fractional, is an NP-hard problem. Their construction uses a polyhedron with exponentially many 0/1-vertices, and the matrix defining the facet inequalities of the polyhedron has exactly two ones per column. In contrast, our construction gives a similar result, but with the further restriction that the polyhedron is *integral*, has only polynomially many 0/1-vertices, and the matrix defining the facet inequalities of the polyhedron is of the form $\left[\frac{A}{r}\right]$, where A is a totally unimodular matrix having at most one "$+1$" and one "-1" per column, and r is a row of $\pm 1'$s.

Another consequence of our construction is that checking if a polyhedron is *half-integral*, i.e., if all the vertices have components in $\{0, 1, 1/2\}$ is an NP-hard problem.

For a polyhedron $P = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$, the support of a vertex $x \in \mathcal{V}(P)$ is defined as the number of positive components of x. Finding a vertex of a polyhedron with maximum support includes several interesting problems, such as MAX-CUT in undirected graphs, and LONGEST-CYCLE in directed graphs. It follows from our construction that it is NP-hard to approximate such maximum support within a factor bigger than $12/n$.

3.2 The polyhedron of negative-weight flows

Given a directed graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}$ on its arcs, consider the following polyhedron:

$$P(G,w) = \left\{ y \in \mathbb{R}^E \;\middle|\; \begin{array}{lrcll} (F) & \displaystyle\sum_{v:(u,v)\in E} y_{uv} - \sum_{v:(v,u)\in E} y_{vu} & = & 0 & \forall \ u \in V \\ (N) & \displaystyle\sum_{(u,v)\in E} w_{uv} y_{uv} & = & -1 & \\ & y_{uv} \geq 0 & & & \forall \ (u,v) \in E \end{array} \right\}.$$

If we think of $w_{u,v}$ as the cost/profit paid for edge (u, v) per unit of flow, then each point of $P(G, w)$ represents a *negative-weight circulation* in G, i.e., assigns a non-negative flow on the arcs, obeying the *conservation of flow* at each node of G, and such that total weight of the flow is strictly negative.

A negative- (respectively, positive-, or zero-) weight cycle in G is a directed cycle whose total weight is negative (respectively, positive, or zero). We represent a cycle C by the subset of arcs appearing on the cycle, and denote by $V(C)$ the nodes of G on the cycle (we assume all cycles considered to be directed and simple). Let us denote the families of all negative, positive, and zero-weight cycles of G by $\mathcal{C}^-(G, w)$, $\mathcal{C}^+(G, w)$, and $\mathcal{C}^0(G, w)$, respectively. Define a *2-cycle* to be a pair of cycles (C_1, C_2) such that $C_1 \in \mathcal{C}^-(G, w), C_2 \in \mathcal{C}^+(G, w)$ and $C_1 \cup C_2$ does not contain any other cycle of G. It is not difficult to see that a 2-cycle is either the edge-disjoint union of a negative cycle C_1 and a positive cycle C_2, or the edge-disjoint union of 3 paths P_1, P_2 and P_3 such that $C_1 = P_1 \cup P_2$ is a negative cycle, and $C_2 = P_1 \cup P_3$ is a positive cycle (see Figure 3.1). In the next section, we show that the vertices $\mathcal{V}(P(G, w))$ are in one-to-one correspondence with the negative cycles $\mathcal{C}^-(G, w)$, while the extreme directions $\mathcal{D}(P(G, w))$ are in one-to-one corre-

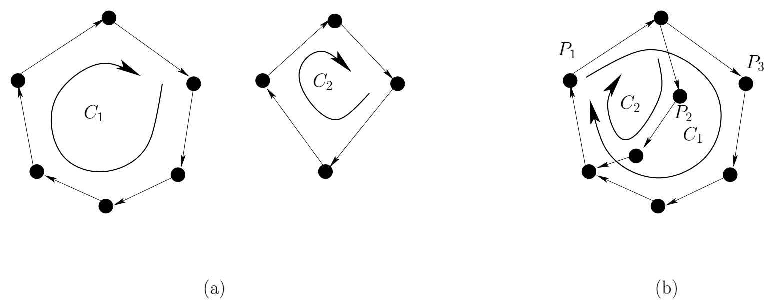


Figure 3.1: 2-cycle.

spondence with the elements of the set $\mathcal{C}^0(G, w) \cup \{(C, C') \ : \ (C, C') \text{ is a 2-cycle}\}$.

3.3 Characterization of vertices and extreme directions of $P(G, w)$

The negative flow polyhedron was also considered in [KBB+06] to obtain the NP-hardness of enumerating vertices of a polyhedron. There it sufficed to show that the vertices of this polyhedron have one-to-one correspondence with the negative cycles of the underlying graph. But to strengthen this result to show that enumerating vertices remains hard for a polyhedron with only 0/1 vertices, one needs to relate the exact coordinate values of the vertices with the weights of the negative cycles. In this section we characterize the extreme rays as a combination of positive and negative cycles and describe the coordinates of the vertices and the extreme rays in terms of the weights of the cycles in the underlying graph.

For a subset $X \subseteq E$, and a weight function $w : E \mapsto \mathbb{R}$, we denote by $w(X) = \sum_{e \in X} w_e$, the total weight of X. For $X \subseteq E$, we denote by $\chi(X) \in \{0, 1\}^E$ the characteristic vector of X: $\chi_e(X) = 1$ if and only if $e \in X$, for $e \in E$.

Theorem 3.3.1. *Let $G = (V, E)$ be a directed graph and $w : E \to \mathbb{R}$ be a real weight on the arcs. Then*

$$\mathcal{V}(P(G, w)) = \left\{ \frac{-1}{w(C)} \chi(C) : \ C \in \mathcal{C}^-(G, w) \right\}, \tag{3.1}$$

$$\mathcal{D}(P(G, w)) = \mathcal{D}_1 \cup \mathcal{D}_2, \tag{3.2}$$

where

$$\begin{aligned}\mathcal{D}_1 &= \{\frac{1}{|C|}\chi(C) :\ C \in \mathcal{C}^0(G,w)\},\\ \mathcal{D}_2 &= \{\mu_{C_1,C_2}\chi(C_1) + \mu'_{C_1,C_2}\chi(C_2)\ :\ (C_1,C_2) \textit{ is a 2-cycle}\},\end{aligned}$$

and

$$\mu_{C_1,C_2} = \frac{w(C_2)}{w(C_2)|C_1| - w(C_1)|C_2|},\ \mu'_{C_1,C_2} = \frac{-w(C_1)}{w(C_2)|C_1| - w(C_1)|C_2|}.$$

are non-negative numbers computed from cycles C_1 and C_2.

Proof. Let $m = |E|$ and $n = |V|$. We first prove (3.1). It is easy to verify that any element $y \in \mathbb{R}^E$ of the set on the right-hand side of (3.1) belongs to $P(G,w)$. Moreover, any such $x = -\chi(C)/w(C)$, for a cycle C, is a vertex of $P(G,w)$ since there are m linearly independent inequalities of $P(G,w)$ tight at x, namely: the conservation of flow equations at $|C|-1$ vertices of C, the equation $\sum_{e\in C} w_e y_e = -1$, and $m - |C|$ equations $y_e = 0$, for $e \in E \setminus C$.

To prove the opposite direction, let $y \in \mathbb{R}^E$ be a vertex of $P(G,w)$. Let $Y = \{e \in E :\ y_e > 0\}$. The proof follows from the following 3 claims.

Claim 1. *The graph (V,Y) is the disjoint union of strongly connected components.*

Proof. Consider an arbitrary strongly connected component X in this graph, and let X^- be the set of components reachable from X (including X). Summing the conservation of flow equations corresponding to all the nodes in X^- implies that all arcs going out of X^- have a flow of zero. □

Claim 2. *There exists no cycle $C \in \mathcal{C}^0(G,w)$ such that $C \subseteq Y$.*

Proof. If such a C exists, we can define two points y' and y'' as follows.

$$y'_e = \begin{cases} y_e + \epsilon, & \text{if } e \in C \\ y_e, & \text{otherwise,} \end{cases} \qquad y''_e = \begin{cases} y_e - \epsilon, & \text{if } e \in C \\ y_e, & \text{otherwise,} \end{cases}$$

for some sufficiently small $\epsilon > 0$. Then y', y'' clearly satisfy (F). Moreover, (N) is satisfied with y' since

$$\sum_{e\in E} w_e y'_e = \sum_{e\notin C} w_e y_e + \sum_{e\in C} w_e(y_e + \epsilon) = \sum_{e\in E} w_e y_e + w(C)\epsilon = -1.$$

Similar argument shows that y'' also satisfies (N). Thus $y', y'' \in P(G, w)$ and $y = (y' + y'')/2$ contradicting that y is a vertex. $\square$

Claim 3. *There exist no distinct cycles $C_1, C_2 \in \mathcal{C}^-(G, w) \cup \mathcal{C}^+(G, w)$ such that $C_1 \cup C_2 \subseteq Y$.*

Proof. If such C_1 and C_2 exist, we can define two points y' and y'' as follows.

$$y'_e = \begin{cases} y_e + \epsilon_1, & \text{if } e \in C_1 \setminus C_2 \\ y_e + \epsilon_2, & \text{if } e \in C_2 \setminus C_1 \\ y_e + \epsilon_1 + \epsilon_2, & \text{if } e \in C_1 \cap C_2 \\ y_e, & \text{otherwise,} \end{cases} \qquad y'_e = \begin{cases} y_e - \epsilon_1, & \text{if } e \in C_1 \setminus C_2 \\ y_e - \epsilon_2, & \text{if } e \in C_2 \setminus C_1 \\ y_e - \epsilon_1 - \epsilon_2, & \text{if } e \in C_1 \cap C_2 \\ y_e, & \text{otherwise,} \end{cases}$$

where $\epsilon_1 = -\frac{w(C_2)}{w(C_1)}\epsilon_2$, for some sufficiently small $\epsilon_2 > 0$ (in particular, to insure non-negativity of y', y'', ϵ_2 must be upper bounded by the minimum of $\min\{y_e : e \in C_2 \setminus C_1\}$, $\frac{|w(C_1)|}{|w(C_2)|}\min\{y_e : e \in C_1 \setminus C_2\}$, and $\frac{|w(C_1)|}{|w(C_1) - w(C_2)|}\min\{y_e : e \in C_1 \cap C_2\}$). Then it is easy to verify that y', y'' satisfy (F). Moreover, (N) is satisfied with y'since

$$\begin{aligned} \sum_{e \in E} w_e y'_e &= \sum_{e \notin C_1 \cup C_2} w_e y_e + \sum_{e \in C_1 \setminus C_2} w_e(y_e + \epsilon_1) + \sum_{e \in C_2 \setminus C_1} w_e(y_e + \epsilon_2) \\ &+ \sum_{e \in C_1 \cap C_2} w_e(y_e + \epsilon_1 + \epsilon_2) = \sum_{e \in E} w_e y_e + w(C_1)\epsilon_1 + w(C_2)\epsilon_2 = -1. \end{aligned}$$

Similarly (N) is also satisfied by y''. Thus $y', y'' \in P(G, w)$ and $y = (y' + y'')/2$ contradicting that y is a vertex of $P(G, w)$. $\square$

The above 3 claims imply that the graph (V, Y) consists of a single cycle C and a set of isolated vertices $V \setminus V(C)$. Thus $y_e = 0$ for $e \notin C$. By (F) we get that y_e is the same for all $e \in C$, and by (N) we get that $y_e = -1/w(C)$ for all $e \in C$, and in particular that $C \in \mathcal{C}^-(G, w)$. This completes the proof of (3.1).

We next prove (3.2). It is easy to see that the extreme directions of $P(G, w)$ are in one-to-one correspondence with the vertices of the polytope $P'(G, w)$, obtained from $P(G, w)$ by setting the right-hand side of (N) to 0 and adding the normalization constraint (N') : $\sum_{e \in E} y_e = 1$. In effect, $P'(G, w)$ is obtained by intersecting the cone of all extreme rays of $P(G, w)$ with a suitable hyperplane, giving a polytope in a lower dimensional affine space.

We first note as before that every element of $\mathcal{D}_1 \cup \mathcal{D}_2$ is a vertex of $P'(G,w)$. Indeed, if $y \in \mathcal{D}_2$ is defined by a 2-cycle (C_1, C_2), then there are m linearly independent inequalities tight at y. To see this, we consider two cases: (i) When C_1 and C_2 are edge-disjoint, then there are $|C_1|-1$ and $|C_2|-1$ equations of type (F), normalization equations (N) and (N'), and $m-|C_1|-|C_2|$ non-negativity inequalities for $e \in E \setminus (C_1 \cup C_2)$. (ii) Otherwise, $C_1 \cup C_2$ consists of 3 disjoint paths P_1, P_2, P_3 of, say m_1, m_2 and m_3 arcs, respectively. Then $C_1 \cup C_2$ has $m_1+m_2+m_3-1$ nodes giving $m_1+m_2+m_3-2$ linearly independent equation of type (F), which together with (N), (N') and $m-m_1-m_2-m_3$ non-negativity constraints for $e \in E \setminus (C_1 \cup C_2)$ uniquely define y.

Consider now a vertex y of $P'(G,w)$. Let $Y = \{e \in E : \; y_e > 0\}$. Clearly, Claim 1 is still valid for Y. On the other hand, Claims 2 and 3 can be replaced by the following two claims.

Claim 4. *There exist no 3 distinct cycles C_1, C_2, C_3 such that $C_1 \in \mathcal{C}^-(G,w)$, $C_2 \in \mathcal{C}^+(G,w)$, and $C_1 \cup C_2 \cup C_3 \subseteq Y$.*

Proof. If such C_1, C_2 and C_3 exist, we can define two points y' and y'' as follows: $y'_e = y_e + \sum_{i=1}^{3} \epsilon_i \chi_e(C_i)$ and $y''_e = y_e - \sum_{i=1}^{3} \epsilon_i \chi_e(C_i)$, for $e \in E$, where $\epsilon_3 > 0$ is sufficiently small, and ϵ_1 and ϵ_2 satisfy

$$\begin{aligned} \epsilon_1 w(C_1) + \epsilon_2 w(C_2) &= -\epsilon_3 w(C_3) \\ \epsilon_1 |C_1| + \epsilon_2 |C_2| &= -\epsilon_3 |C_3|. \end{aligned} \tag{3.3}$$

Note that ϵ_1 and ϵ_2 exist since $\alpha \overset{\text{def}}{=} w(C_1)|C_2| - w(C_2)|C_1| < 0$. Furthermore, since $\epsilon_1 = (w(C_2)|C_3| - w(C_3)|C_2|)\epsilon_3/\alpha$ and $\epsilon_2 = (w(C_3)|C_1| - w(C_1)|C_3|)\epsilon_3/\alpha$, we can select ϵ_3 such that $y', y'' \geq 0$. By definition of y' and y'', they both satisfy (F), and by (3.3) they also satisfy (N) and (N'). However, $(y'+y'')/2 = y$ which contradicts that $y \in \mathcal{V}(P'(G,w))$. □

Claim 5. *There exist no 2 distinct cycles C_1, C_2 such that $C_1, C_2 \in \mathcal{C}^0(G,w)$, and $C_1 \cup C_2 \subseteq Y$.*

Proof. If such C_1 and C_2 exist, we define two points y' and y'' as follows: $y'_e = y_e + \epsilon_1 \chi_e(C_1) + \epsilon_2 \chi_e(C_2)$ and $y''_e = y_e - \epsilon_1 \chi_e(C_1) - \epsilon_2 \chi_e(C_2)$, for $e \in E$, where $\epsilon_2 > 0$ is sufficiently small, and $\epsilon_1 = -\epsilon_2 |C_2|/|C_1|$. Then $y', y'' \in P'(G,w)$ and $y = (y'+y'')/2$. □

As is well-known, we can decompose y into the sum of positive flows on cycles, i.e., write $y = \sum_{C \in \mathcal{C}'} \lambda_C \chi(C)$, where $\mathcal{C}' \subseteq \mathcal{C}^-(G,w) \cup \mathcal{C}^+(G,w) \cup \mathcal{C}^0(G,w)$, and $\lambda_C > 0$ for $c \in \mathcal{C}'$. It follows from Claim 4 that $|\mathcal{C}'| \leq 2$. Using (N), we get $\sum_{c \in \mathcal{C}'} \lambda_C w(C) = 0$, which implies by Claim 5 that either $\mathcal{C}' = \{C\}$ and $w(C) = 0$ or $\mathcal{C}' = \{C_1, C_2\}$ and $w(C_1) < 0$, $w(C_1) > 0$. In the former case, we get that $y \in \mathcal{D}_1$, and in the latter case, we get by Claim 4 that (C_1, C_2) is a 2-cycle, and hence, that $y \in \mathcal{D}_2$. $\square$

In the next section we construct a weighted directed graph (G, w) in which all negative cycles have unit weight. We show that generating all negative cycles of G is NP-hard, thus implying by Theorem 3.3.1 that generating all vertices of $P(G, w)$ is also hard.

3.4 NP-hardness construction

In this section we will describe a reduction from 3-SAT to the problem of enumerating the negative cycles of a directed graph. The construction is essentially the same as in [KBB+06]; only the weights change. Note however that the weights used in [KBB+06] are symmetric while we use asymmetric weights. This asymmetric weighting is crucial to ensure that all the negative cycles have the same weight and hence all the vertices of the resulting polyhedron are 0/1.

The 3SAT problem is the following decision problem: Given a CNF boolean formula $\phi = C_1 \wedge \cdots \wedge C_M$ on N variables $x_1, \cdots, x_N$ such that every clause C_i contains exactly 3 literals, is ϕ satisfiable?

For any given 3SAT formula ϕ, we will construct a directed graph $G(\phi)$ with the following property. $G(\phi)$ has a number of *short* negative cycles that is polynomial in the size of ϕ and such short cycles can be easily enumerated. Furthermore, $G(\phi)$ has *long* negative cycles if and only if ϕ is satisfiable. In the context of this reduction, a cycle is called short if it has only two nodes and long otherwise.

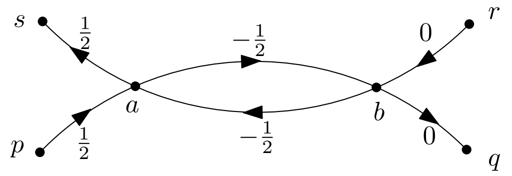


Figure 3.2: Paths for a literal occurrence

We distinguish each occurrence of a literal and denote an occurrence of x_i in C_j as x_i^j. For every x_i^j we introduce two paths (see Figure 3.2) on six vertices p, q, a, b, r, s and six edges. The directed edges are added to define the directed paths p, a, b, q and r, b, a, s. The added edges have the following weights:

$$w(p,a) = \tfrac{1}{2}, \quad w(a,b) = -\tfrac{1}{2}, \quad w(b,q) = 0,$$

$$w(r,b) = 0, \quad w(b,a) = -\tfrac{1}{2}, \quad w(a,s) = \tfrac{1}{2}.$$

It is useful to think of these two paths as separate parts of the graph as depicted in Figure 3.3. with the nodes a, b identified with a', b' respectively. For a literal-occurrence x_i^j, we will call the path containing a, b as $\mathcal{P}(x_i^j)$ and the path containing a', b' as $\mathcal{P}'(x_i^j)$. Now, for each literal x_i we create a gadget $\mathcal{G}_i$ which consists of two parallel paths - one corresponding to the positive occurrences of x_i in any clause and the other corresponding to the negative occurrences. To get each of these parallel paths we simply concatenate the various paths $\mathcal{P}(x_i^j)$.

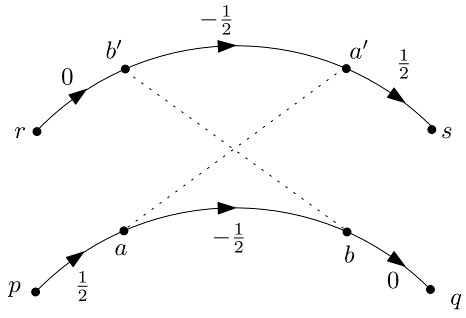


Figure 3.3: The dotted lines indicate the nodes that are identified as one node.

To give an example, let $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$ be the given 3SAT formula. The literal x_1 appears in C_1 and C_2 in positive form and is negated in C_3. Therefore, we concatenate $\mathcal{P}(x_1^1)$ and $\mathcal{P}(x_1^2)$ to obtain one path in $\mathcal{G}_1$ and $\mathcal{P}(\overline{x}_1^3)$ becomes the other parallel path in $\mathcal{G}_1$. The gadget G_1 for this example is shown in Figure 3.4.

Now, for every clause C_i we construct a gadget $\mathcal{G}'_i$ as follows. $\mathcal{G}'_i$ consists of three parallel paths - each corresponding to the path $\mathcal{P}'$ of one of the literals appearing in C_i. For the clause C_1 in our example formula ϕ the gadget $\mathcal{G}'_1$ is shown again in Figure 3.5.

For each of the gadgets $\mathcal{G}_i$ and $\mathcal{G}'_i$ there is a unique source and a unique sink.

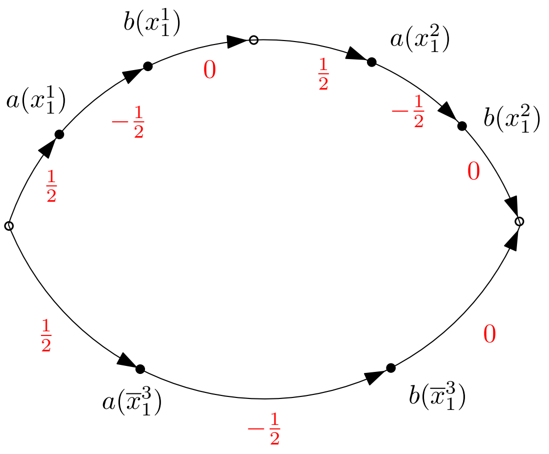


Figure 3.4: Gadget for x_1 in $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$

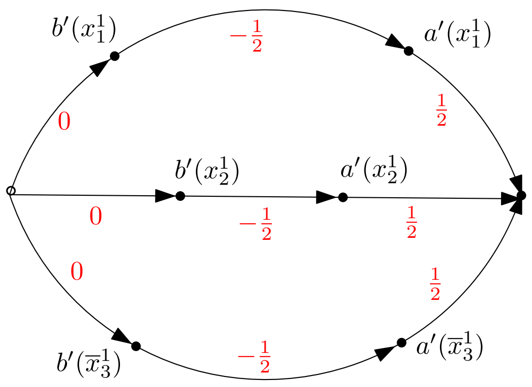


Figure 3.5: Gadget for C_1 in $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$

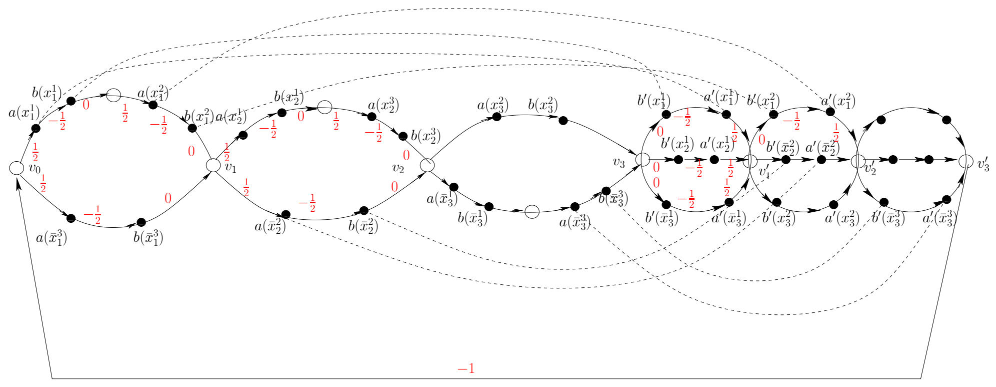


Figure 3.6: An example of the graph construction in the proof of Theorem 3.5.1 with $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$.

We concatenate gadget $\mathcal{G}_i$ with $\mathcal{G}_{i+1}$, for $i \in \{1, \cdots, N-1\}$, by simply identifying the sink of $\mathcal{G}_i$ with the source of $\mathcal{G}_{i+1}$. This gives us one gadget $\mathcal{G}$ for the literals. We similarly concatenate the gadgets $\mathcal{G}'_i$ together to get a gadget $\mathcal{G}'$ for the clauses. Next we identify the sink of $\mathcal{G}$ with the source of $\mathcal{G}'$ and add a directed edge from the sink of $\mathcal{G}'$ to the source of $\mathcal{G}$. This new edge is given the weight $-h$. The precise value of $h > 0$ will be specified later.

The final graph that we obtain can be thought of as a sequence of parallel chains joined together as follows (see Figure 3.6 for the graph resulting from our running example ϕ):

$$G = v_0\ \mathcal{G}_1\ v_1\ \mathcal{G}_2\ v_2 \ldots\ v_{N-1}\ \mathcal{G}_n\ v_N\ \mathcal{G}'_1\ v'_1\ \mathcal{G}'_2\ v'_2 \ldots\ v'_{M-1}\ \mathcal{G}'_M\ v'_M,$$

where $v_0, v_1, \ldots, v_N, v'_1, \ldots, v'_{M-1}, v'_M$ are distinct vertices, each $\mathcal{G}_i$, for $i = 1, \ldots, N$, corresponds to the literal x_i, and each $\mathcal{G}'_j$, for $j = 1, \ldots, M$, corresponds to the clause C_j. Recall that the nodes $a(\ell)$ and $b(\ell)$ for a literal ℓ in $\mathcal{G}$ are identified with the nodes $a'(\ell)$ and $b'(\ell)$ in $\mathcal{G}'$. The dotted lines in the Figure 3.6 connect nodes that are identified as the same nodes.

Clearly the arcs $(a(\ell), b(\ell))$ and $(b'(\ell), a'(\ell))$ form a directed cycle of total weight -1, for every literal occurrence ℓ. Let $\mathcal{S} \subseteq \mathcal{C}^-(G, w)$ be the set of such *short* cycles. Note that $|\mathcal{S}| = \sum_{j=1}^{M} |C_j| = 3M$.

Call a cycle of G *long* if it contains the vertices $v_0, v_1, \ldots, v_N, v'_1, \ldots, v'_{M-1}, v'_M$. Any long cycle has weight $-h$.

Remark Every long cycle contains the directed edge (v'_M, v_0), but no short negative cycle contains this edge.

The crucial observation is the following.

Claim 6. *Any negative cycle $C \in \mathcal{C}^-(G,w) \setminus \mathcal{S}$ must either be long or have weight at least $-h+1$.*

Proof. Consider any cycle $C \notin \mathcal{S}$, and let us write the traces of the nodes visited on the cycle (dropping the literals, and considering a, a' and b, b' as different copies), without loss of generality as follows:

$$p\ a\ b\ p\ a\ b\ p\ \cdots\ a\ a'\ s\ b'\ a'\ s\ b'\ \cdots\ b'\ b\ p\ a\ b\ \cdots\ p.$$

Note that the sequences $a'\ a$ and $b\ b'$ are not allowed since otherwise C contains a cycle from $\mathcal{S}$.

Let us compute the distance (i.e., the total weight) of each node on this sequence starting from the initial node p. Call the subsequences $a\ a'$ and $b\ b'$, a- and b-jumps respectively. Then it is easy to verify that each a-jump causes the distance to eventually increase by 1 while each b-jump keeps the distance at its value. More precisely, the distance at a node x in the sequence is given by $d(x) = t(x) + d_0(x) - \delta(x)$, where t is the number of a-jumps appearing upto x, and

$$d_0(x) = \begin{cases} 0 & \text{if } x \in \{p, s\}, \\ \frac{1}{2} & \text{if } x = a, \\ -\frac{1}{2} & \text{if } x = a', \\ 0 & \text{if } x = b = b', \end{cases}$$

$$\delta(x) = \begin{cases} h & \text{if arc } (v'_M, v_0) \text{ appears on the path from } p \text{ to } x \\ 0 & \text{otherwise.} \end{cases}$$

One also observes that, if the sequence has a b-jump, and it contains the nodes v_0 and v'_M, then it must also contain an a-jump. Thus it follows from the definition of $d(x)$ that any cycle with a jump must be either non-negative, if it does not contain the nodes v_0 and v'_M, or have weight at least $-h+1$, if it contains v_0 and v'_M. So the only possible negative cycle, not in $\mathcal{S}$, of weight less than or equal to $-h$ must be long. □

We summarize this construction in form of the following lemma:

Lemma 3.4.1. *For any* 3*-CNF* ϕ *with* m *clauses we can obtain a graph* $G(\phi)$ *with* $18m + 1$ *edges such that* $G(\phi)$ *has* $3m$ *short negative cycles. Furthermore, every negative cycle in* $G(\phi)$ *has total weight* -1*.*

Let (G, w) be the graph with weights defined as above. Also, let $\mathcal{S}$ be the set of all short negative cycles and $\mathcal{C}^-(G, w)$ be the set of all negative cycles in G. The following claim was established in [KBB+06].

Lemma 3.4.2. *The CNF formula* ϕ *is satisfiable if and only if* $\mathcal{S} \subset \mathcal{C}^-(G, w)$*.*

We omit the proof of this Lemma here because even with the new weights that we use, the proof remains exactly the same.

3.5 Hardness of checking some polyhedral properties

Let $P(G(\phi), w)$ be the polyhedron defined by the graph $G(\phi)$ and the arc weights w, constructed for an input CNF formula ϕ as in the previous section. Let $\mathcal{X} \subseteq \mathcal{V}(P(G(\phi), w))$ be the vertices of $P(G(\phi), w)$ corresponding to the set $\mathcal{S} \subseteq \mathcal{C}^-(G(\phi), w)$, defined in the previous section.

3.5.1 Generating all vertices of a $0/1$-polyhedron is hard

Let us now show that the following problem is coNP-complete:

VE-$0/1$**:** Given a polyhedron $P = \{x \in \mathbb{R}^n |\ Ax \le b\}$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\mathcal{V}(P) \subseteq \{0, 1\}^n$, and a subset $\mathcal{X} \subseteq \mathcal{V}(P)$, decide if $\mathcal{X} = \mathcal{V}(P)$.

A consequence of the coNP-completeness of the above problem is that there is no output-sensitive algorithm to enumerate the elements of $\mathcal{V}(P)$, unless $P = NP$.

Theorem 3.5.1. *Problem VE-*$0/1$ *is NP-hard.*

Proof. Set $h = 1$ in the construction. Then by Claim 6, any negative cycle must either belong to $\mathcal{S}$ or is long. Any such cycle has weight -1. This implies by

Theorem 3.3.1 that all vertices of $P(G, w)$ are 0/1, and further that checking if $\mathcal{V}(P(G(\phi), w)) = \mathcal{X}$ is equivalent to checking if $\mathcal{C}^-(G(\phi), w) = \mathcal{S}$. The latter problem is NP-hard by Lemma 3.4.2. $\square$

Thus, it is NP-hard to generate all vertices of a 0/1-polyhedron. However, (3.2) shows that the above construction cannot be used to imply the same hardness result for polytopes, since the numbers of positive and negative cycles can be exponential. In fact, for the negative cycle polyhedron arising in the construction of Theorem 3.5.1, we have the following.

Proposition 3.5.2. *For the directed graph $G = (V, E)$ and weight $w : E \to \mathbb{R}$ used in the proof of Theorem 3.5.1, both sets $\mathcal{D}(P(G, w))$ and $\mathcal{V}(P(G, w)) \cup \mathcal{D}(P(G, w))$ can be generated in output-sensitive polynomial time.*

Proof. This follows from the fact that for every positive cycle in G there is a negative cycle, edge-disjoint from it, and vice versa (assuming no clause consists of only one literal). Hence, the number of 2-cycles and thus the number of extreme directions of $P(G, w)$ satisfy $|\mathcal{D}(P(G, w))| \geq \max\{|\mathcal{C}^+(G, w)|, |\mathcal{C}^-(G, w)|\} + |\mathcal{C}^0(G, w)|$. Thus $\mathcal{D}(P(G, w)$ and $\mathcal{V}(P(G, w)) \cup \mathcal{D}(P(G, w))$ can be generated by generating all cycles of G, which can be done efficiently [RT75]. $\square$

However, we do not know if the 2-cycles for general graphs can be efficiently enumerated. In fact, there exist weighted graphs in which the number of positive cycles is exponentially larger than the number of 2-cycles. Consider for instance, a graph G composed of a directed cycle $(x_1, y_1, \ldots, x_k, y_k)$ of length $2k$, all arcs with weight -1, and $2k$ additional paths $\mathcal{P}_1, \mathcal{P}'_1, \ldots, \mathcal{P}_k, \mathcal{P}'_k$ where $\mathcal{P}_i = (x_i, z_i, y_i)$ and $\mathcal{P}'_i = (x_i, z'_i, y_i)$, of two arcs each going the same direction parallel with every second arc along the cycle, each having a weight of $2k$ (see Figure 3.7 for an example with $k = 4$). Then we have more than 2^k positive cycles, but the number of 2-cycles is only $2k$. For such graphs enumerating all cycles and then combining the positive and the negative cycles to enumerate all 2-cycles is clearly not output-sensitive. Note that proving the NP-hardness of enumerating 2-cycles of a given weighted will imply the same for the vertex enumeration problem for polytopes.

3.5.2 Recognizing 0/1-polyhedra is hard

The problem of checking if a polyhedron $P = \{x \in \mathbb{R}^n : \ Ax \geq \mathbf{e}, \ x \geq 0\}$, where $A \in \{0, 1\}^{V \times E}$ is the vertex-edge incidence matrix of an undirected graph

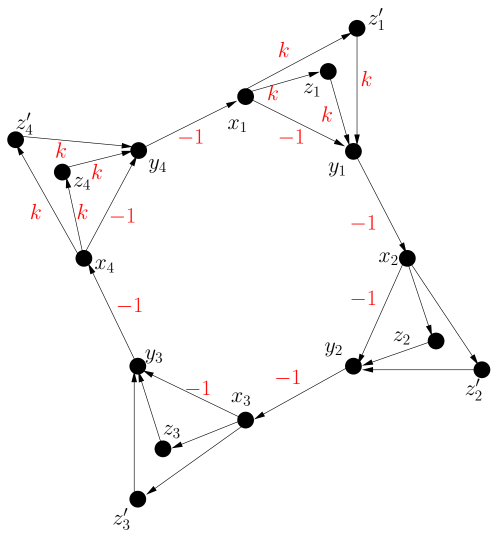


Figure 3.7: An example where there are exponentially more positive cycles than 2-cycles ($k = 4$).

$G = (V, E)$ and $\mathbf{e}$ is the vector of all ones, is a 0/1 polyhedron, was shown in [DFZ07] to be NP-hard. The 0/1-vertices of P are the *edge-covers* of G, whose number is exponential in n for the graph used in the NP-hardness construction of [DFZ07], and these are the only vertices of P which could possibly be integral. In other words, the result of [DFZ07] implies that is hard to distinguish if a polyhedron is 0/1 or fractional. In contrast, we show here the following.

Theorem 3.5.3. *Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, such that $A = \left[\frac{A'}{r}\right]$ has exactly one "+1" and one "-1" per column, $r \in \{-1, +1\}^m$, $b = [0, \ldots, 0, -2]^T$, and $\mathcal{V}(P) \subseteq \{0, 1, 2\}^n$, it is NP-hard to tell if $\mathcal{V}(P) \subseteq \{0, 1\}^n$, even if P has a polynomial number of 0/1-vertices.*

Proof. We set $h = \frac{1}{2}$. It follows that all short negative cycles have weight -1, any long negative cycle has weight $-1/2$, and by Claim 6, there exists no other negative cycles. In particular, by (3.1), $\mathcal{V}(P(G, w))$ can be partitioned into two sets $\mathcal{V}_1, \mathcal{V}_2$, where $\mathcal{V}_1$ is the set of vertices corresponding to short negative cycles, and $\mathcal{V}_2$ are the ones corresponding to long negative cycles. The theorem follows from the following facts

(i) $P(G, w)$ is integral: $\mathcal{V}(P(G, w)) = \mathcal{V}_1 \cup \mathcal{V}_2 \subseteq \{0, 1, 2\}^E$,

(ii) $\mathcal{V}_1 \subseteq \{0, 1\}^E$ and $\mathcal{V}_2 \subseteq \{0, 2\}^E$,

(iii) $|\mathcal{V}_1| \leq |E|$, and

(iv) checking if $\mathcal{V}_2$ is non-empty is NP-hard.

□

Papadimitriou and Yannakakis [PY90] showed that checking if a polytope, given by its facets, is integral is an NP-hard problem. It will be interesting to extend their result by showing that checking if a given integral polytope is 0/1 is also hard.

3.5.3 Checking for half-integrality is hard

Let f be an integer. A polyhedron $P \subseteq \mathbb{R}^n$, with $\mathcal{V}(P) \in [0,1]^n$, is said to be $\frac{1}{f}$-*integral* [Vaz01] if $\mathcal{V}(P) \subseteq \{0, \frac{1}{f}, \frac{1}{f-1}, \ldots, \frac{1}{2}, 1\}^n$. In particular, for $f = 1/2$, such a polyhedron is called *half-integral.* For instance, if $A \in \{0,1\}^{E \times V}$ is the edge-vertex incidence matrix of a graph $G = (V, E)$, then the polyhedron $P = \{x \in \mathbb{R}^E \ : \ Ax \geq \mathbf{e},\ x \geq 0\}$ is half-integral. The importance of half-integral (or more generally $\frac{1}{f}$-integral) polyhedra $P = \{x \in \mathbb{R}^n : \ Ax = b,\ x \geq 0\}$ is that, for any $c \in \mathbb{R}^n$, one can approximate the optimum of $\min\{cx \ : \ x \in P \cap \{0,1\}^n\}$ within a relative factor of 2, by solving the linear programming relaxation $\min\{cx \ : \ x \in P\}$, and rounding to 1 all variables with value at least $1/2$. Thus it will be interesting to be able to recognize such classes of polyhedra in polynomial time. The next result states that this is highly unlikely, unless P=NP.

Theorem 3.5.4. *Given a polyhedron P, with $\mathcal{V}(P) \subseteq [0,1]^n$ and an integer $f \geq 1$. It is NP-hard to decide if P is $\frac{1}{f}$-integral.*

Proof. We set $h = (f+1)$ in the construction. Then Claim 6 implies that any non-long negative cycle has weight of at least $-f$, while a long negative cycle has weight $-(f+1)$. It follows that each vertex of $P(G, w)$ corresponding to a non-long negative cycle has components in $\{0, 1, \frac{1}{2}, \ldots, \frac{1}{f}\}$. Thus the only negative cycle corresponding to a vertex with some component possibly less than $1/f$ is a long negative cycle. But checking for the existence of such cycle is NP-hard by Lemma 3.4.2. □

3.5.4 Hardness of approximating the maximum support

For a polyhedron $P = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$, where $A \in \mathbb{R}^{m \times n}$, the support of a vertex $x \in \mathcal{V}(P)$ is defined as $\text{supp}(x) = |\{i \in [n] \ : \ x_i > 0\}|$ the number of positive components of x. Let $\text{max-supp}(P) = \max\{\text{supp}(x) \ : \ x \in \mathcal{V}(P)\}$. Given an unweighted directed graph $G = (V, E)$, let us assign weight -1 to each arc. Then $P(G, w)$ is polytope whose vertices are in one-to-one correspondence with the directed simple cycles of G. It follows that the vertex with maximum support in $\mathcal{V}(P)$ corresponds to the longest cycle in G. It was shown in [BHK04] that it is not possible to approximate the longest cycle in directed graph within a factor $|V|^{1-\epsilon}$, for any $\epsilon > 0$, unless $P = NP$. It follows that $\text{max-supp}(P)$, for a polytope P, is NP-hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$. Here we show a stronger result for polyhedra.

Theorem 3.5.5. *For a polyhedron $P = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$, the following problems are NP-hard:*

(i) Checking if P has a vertex of support more than 2.

(ii) Checking if P has a vertex with $x_i > 0$ for a given i.

(iii) Approximating $\text{max-supp}(P)$ within a factor of bigger than $12/n$.

Proof. Set $h = 1$ in the construction of Section 3.4. (i) follows from the observation that vertices of $P(G(\phi), w)$ corresponding to short cycles have support 2, while the existence of a vertex of bigger support is equivalent to the existence of a long cycle. (ii) follows from the observation that a long cycle, if one exists, must contain the arc (v'_M, v_0). To see (iii), observe that the hardness construction remains valid even if we assume that the CNF formula ϕ has 3 literals per clause, and each literal appears in at least one of the clauses. With such an assumption and using the notation of Section 3.4, we have $n = |E| = 18M + 1$. Any long cycle has length at least $3(N + M) + 1 \geq n/6$ while a short cycle has length 2. Thus we can check the existence of a long cycle if we cannot approximate $\text{max-supp}(P(G(\phi), w))$ within a factor bigger than $\frac{12}{n}$.

□

Chapter 4

Centroid of a Polytope

4.1 Introduction

Let $\mathcal{P}$ be an $\mathcal{H}$-polytope in $\mathbb{R}^d$ with vertex set V. Various notions try to capture the essence of a "center" of a polytope. Perhaps the most popular notion is that of the center of mass of $\mathcal{P}$. Formally, the center of mass, $c_\mu(S)$ of a full-dimensional body $S \subset \mathbb{R}^d$ is defined as follows

$$c_\mu(S) = \frac{\int_{x \in S} x \, dx}{\int_{x \in S} dx}$$

Recently Rademacher proved that computing the center of mass of a polytope is #P-hard [Rad07]. The proof essentially relies on the fact that the center of mass captures the volume of a polytope perfectly and that computing the volume of a polytope is #P-hard [DF88]. Note that, polynomial algorithms exist that approximate the volume of a polytope within any arbitrary factor [KLS98]. It is also easy to see that the center of mass can be approximated by simply sampling random points from the polytope, the number of samples depending polynomially on the desired approximation (See Algorithm 5.8 of [KLS98]).

In this chapter we study a variant of the notion of "center" defined as the centroid (average) of the vertices of P. Despite being quite a natural feature of polytopes, this variant seems to have received very little attention both from theoretical and computational perspectives. Throughout this chapter we will refer to the vertex centroid just as centroid. The reader should note that in popular literature the word centroid refers more commonly to the center of mass. We nevertheless use the same terminology for simplicity of language. Our motivation for studying the centroid stems from the fact that the centroid encodes the number of vertices of a polytope. As we will see, this also makes computing the centroid hard.

The parallels between centroid and the center of mass of a polytope mimic the parallels between the volume and the number of vertices of a polytope. Computing the volume and the number of vertices are both #P-complete ([Dye83, DF88, Lin86]) and so are the problems of computing the corresponding centroids ([Rad07], Theorem 4.2.1). The volume can be approximated quite well but approximating the number of vertices of a polytope is an interesting open problem. Similarly, the center of mass can be approximated quite well but (as we will see in this chapter) obtaining a polynomial algorithm for approximating the centroid would be a very interesting achievement.

The main results of this chapter are the following:

- Computing the centroid of an $\mathcal{H}$-polytope is #P-hard.
- Even just deciding whether the centroid of an $\mathcal{H}$-polytope lies in a halfspace remains #P-hard.
- Approximating the centroid of an $\mathcal{H}$-polytope is #P-easy.
- Any algorithm approximating the centroid of any polytope, contained in the unit hypercube, within a distance $d^{\frac{1}{2}-\delta}$ for any fixed constant $\delta > 0$ can be used to obtain a fully polynomial approximation scheme for the centroid approximation problem and also an output sensitive polynomial algorithm for the Vertex Enumeration problem.
- There is no polynomial algorithm that approximates the vertex centroid of arbitrary $\mathcal{H}$-polyhedron within a distance $d^{\frac{1}{2}-\delta}$ for any fixed constant $\delta > 0$, unless $P = NP$.

We should remark that for the approximation of centroid, we only consider polytopes (and polyhedra) whose vertices lie inside a unit hypercube. To see how this assumption can easily be satisfied, notice that a halfspace h can be added to a polyhedron P such that $P \cap h$ is bounded and the vertices of P are preserved in $P \cap h$. Also, such a halfspace can be found in polynomial time from the inequalities defining P. Once we have a polytope in $\mathbb{R}^d$, solving $2d$ linear programs gives us the width along each coordinate axis. The polytope can be scaled by a factor depending on the width along each axis to obtain a polytope all whose vertices lie inside a unit hypercube. In case we started with a polyhedron P, the scaled counterpart of the halfspace h that was added can be thrown to get back a polyhedron that is a scaled version of P and all whose vertices lie inside the unit hypercube. In section 4.3 we provide further motivation for this assumption.

Since all the vertices of the polytope (or polyhedron) lie inside a unit hypercube, picking any arbitrary point from inside this hypercube yield a $d^{\frac{1}{2}}$-approximation of the vertex centroid. Thus, the last result should be contrasted to the fact that approximating the vertex centroid within a distance of $d^{\frac{1}{2}}$ is trivial. Also, even though we discuss only polytopes *i.e.* bounded polyhedra in sections 4.2 and 4.3, the results and the proofs are valid for the unbounded case as well. We discuss the unbounded case explicitly only in section 4.4.

4.2 Exact Computation of the Centroid

The most natural computational question regarding the centroid of a polytope is whether we can compute the centroid efficiently. The problem is trivial if the input polytope is presented by its vertices. So we will assume that the polytope is presented by its facets. Perhaps not surprisingly, computing the centroid of an $\mathcal{H}$-polytope turns out be #P-hard. We prove this by showing that computing the centroid of an $\mathcal{H}$-polytope amounts to counting the vertices of the same polytope, a problem known to be #P-hard.

Theorem 4.2.1. *Given an $\mathcal{H}$-polytope $\mathcal{P} \subset \mathbb{R}^d$, it is #P-hard to compute its centroid $c(\mathcal{P})$.*

Proof. Embed $\mathcal{P}$ in $\mathbb{R}^{d+1}$ by putting a copy of $\mathcal{P}$ in the hyperplane $x_{d+1} = 1$ and making a pyramid with the base $\mathcal{P}$ and apex at the origin. Call this new polytope $\mathcal{Q}$. Treating the direction of the positive x_{d+1}-axis as up, it is easy to see that the centroid of the new polytope lies at a height $1 - \frac{1}{n+1}$ iff the number of vertices of $\mathcal{P}$ is n. Thus any algorithm for computing the centroid can be run on $\mathcal{Q}$ and the number of vertices of $\mathcal{P}$ can be read off the $(d+1)$-st coordinate. $\square$

Suppose, instead, that one does not want to compute the centroid exactly but is just interested in knowing whether the centroid lies to the left or to the right of a given arbitrary hyperplane. This problem turns out to be hard too, and it is not difficult to see why.

Theorem 4.2.2. *Given an $\mathcal{H}$-polytope $\mathcal{P} \subset \mathbb{R}^d$ and a hyperplane $h = \{a \cdot x = b\}$, it is #P-hard to decide whether $a \cdot c(\mathcal{P}) \leq b$.*

Proof. Consider the embedding and the direction pointing upwards as used in the proof of Theorem 4.2.1. Given an oracle answering sidedness queries for the centroid and any arbitrary hyperplane, one can perform a binary search on the height of the centroid and locate the exact height. The number of queries needed is only logarithmic in the number of vertices of $\mathcal{P}$. □

4.3 Approximation of the Centroid

As stated before, even though computing the gravitational centroid of a polytope exactly is #P-hard, it can be approximated efficiently to any precision by random sampling. Now we consider the problem of similarly approximating the vertex centroid of an $\mathcal{H}$-polytope. Let $dist(x, y)$ denote the Euclidean distance between two points $x, y \in \mathbb{R}^d$. We are interested in the following problem:
Input: $\mathcal{H}$-polytope $P \subset \mathbb{R}^d$ and a real number $\epsilon > 0$.
Output: $p \in \mathbb{R}^d$ such that $dist(c(P), p) \leq \epsilon$.

We would like an algorithm for this problem that runs in time polynomial in the number of facets of P, the dimension d and $\frac{1}{\epsilon}$. Clearly, such an algorithm would be very useful because if such an algorithm is found then it can be used to test whether a polytope described by m facets has more than n vertices, in time polynomial in m, n and the dimension d of the polytope by setting $\epsilon < \frac{1}{2}\left(\frac{1}{n} - \frac{1}{n+1}\right)$. This in turn would yield an algorithm that computes the number of vertices n of a d-dimensional polytope with m facets, in time polynomial in m, n and d. As stated before, a problem that is polynomially equivalent to the Vertex Enumeration problem is to decide if a given list of vertices of an $\mathcal{H}$-polytope is complete [ABS97]. Clearly then, a polynomial approximation scheme for the centroid problem would yield an output-sensitive polynomial algorithm for the Vertex Enumeration problem.

Also, the problem of approximating the centroid is not so interesting if we allow polytopes that contain an arbitrarily large ball, since this would allow one to use an algorithm for approximating the centroid with *any* guarantee to obtain another algorithm with an arbitrary guarantee by simply scaling the input polytope appropriately, running the given algorithm and scaling back. So we will assume that the polytope is contained in a unit hypercube in $\mathbb{R}^d$.

Now we prove that the problem of approximating the centroid is #P-easy. We do this by showing that given an algorithm that computes the number of vertices of an arbitrary polytope (a #P-complete problem), one can compute the centroid to any desired precision by making a polynomial (in $\frac{1}{\epsilon}$, the number of facets and the dimension of the polytope) number of calls to this oracle. Notice that in the approximation problem at hand, we are required to find a point within a d-ball centered at the centroid of the polytope and radius ϵ. We first modify the problem a bit by requiring to report a point that lies inside a hypercube, of side length 2ϵ, centered at the centroid of the polytope. (The hypercube has a clearly defined center of symmetry, namely its own vertex centroid.) To see why this does not essentially change the problem, note that the unit hypercube fits completely inside a d-ball with the same center and radius $\frac{\sqrt{d}}{2}$. We will call any point that is a valid output to this approximation problem, an ϵ-approximation of the centroid $c(P)$.

Given an $\mathcal{H}$-polytope P and a hyperplane $\{a \cdot x = b\}$ that intersects P in the relative interior and does not contain any vertex of P, define P_1 and P_2 as follows:

$$\begin{aligned} P_1 &= P \cap \{x | a \cdot x \leq b\} \\ P_2 &= P \cap \{x | a \cdot x \geq b\} \end{aligned}$$

Let V_1 be the common vertices of P_1 and P, and V_2 be common vertices of P_2 and P. The following lemma gives a way to obtain the ϵ-approximation of the centroid of P from the ϵ-approximations of the centroids of V_1 and V_2.

Lemma 4.3.1. *Given P, V_1, V_2 defined as above, let n_1 and n_2 be the number of vertices in V_1 and V_2 respectively. If c_1 and c_2 are ϵ-approximations of the centroids of V_1 and V_2 respectively, then $c = \frac{n_1 c_1 + n_2 c_2}{n_1 + n_2}$ is an ϵ-approximation of the centroid c^* of P.*

Proof. Let c_{ij} be the j-th coordinate of c_i for $i \in \{1, 2\}$. Also, let c_i^* be the actual centroid of V_i with c_{ij}^* denoting the j-th coordinate of c_i^*. Since c_i approximates c_i^* within a hypercube of side-length 2ϵ, for each $j \in \{1, \cdots, d\}$ we have

$$c_{ij}^* - \epsilon \leq \quad c_{ij} \quad \leq c_{ij}^* + \epsilon$$

Also, since c^* is the centroid of P,

$$c^* = \frac{n_1 c_1^* + n_2 c_2^*}{n_1 + n_2}$$

Hence, for each coordinate c_j^* of c^* we have

$$\begin{aligned}
& \tfrac{n_1(c_{1j}-\epsilon)+n_2(c_{2j}-\epsilon)}{n_1+n_2} && \leq c_j^* \leq \frac{n_1(c_{1j}+\epsilon)+n_2(c_{2j}+\epsilon)}{n_1+n_2} \\
\Rightarrow\quad & \tfrac{n_1c_{1j}+n_2c_{2j}}{n_1+n_2} - \epsilon && \leq c_j^* \leq \frac{n_1c_{1j}+n_2c_{2j}}{n_1+n_2} + \epsilon \\
\Rightarrow\quad & c_j - \epsilon && \leq c_j^* \leq c_j + \epsilon \\
\Rightarrow\quad & c_j^* - \epsilon && \leq c_j \leq c_j^* + \epsilon
\end{aligned}$$

□

Now to obtain an approximation of the centroid, we first slice the input polytope P from left to right into $\frac{1}{\epsilon}$ slices each of thickness at most ϵ. Using standard perturbation techniques we can ensure that any vertex of the input polytope does not lie on the left or right boundary of any slice. For each slice any point in the interior gives us an ϵ-approximation of the vertices of P that are contained in that slice. We can compute the number of vertices of P lying in this slice by using the oracle for vertex computation and then using the previous Lemma we can obtain the centroid of P. Thus we have the following theorem:

Theorem 4.3.2. *Given a polytope P contained in the unit hypercube, the ϵ-approximation of the centroid of P can be computed by making a polynomial number of calls to an oracle for computing the number of vertices of a polytope.*

Now we present a bootstrapping theorem indicating that any "sufficiently" non-trivial approximation of the centroid can be used to obtain arbitrary approximations. For the notion of approximation let us revert back to the Euclidean distance function. Thus, any point x approximating the centroid c within a parameter ϵ satisfies $dist(x, c) \leq \epsilon$. As before we assume that the polytope $\mathcal{P}$ is contained in the unit hypercube. Since the polytope is thus contained in a hyperball with origin as its center and radius at most $\frac{\sqrt{d}}{2}$, any point inside $\mathcal{P}$ approximates the centroid within a factor $\sqrt{d}$. Before we make precise our notion of "sufficiently" non-trivial and present the bootstrapping theorem, some preliminaries are in order.

Lemma 4.3.3. *Suppose $(x, y), (u, u) \in \mathbb{R}^{2d}$, where $x, y, u \in \mathbb{R}^d$, then*

$$||u - \frac{x+y}{2}|| \leq \frac{||(u,u) - (x,y)||}{\sqrt{2}},$$

where $|| \cdot ||$ is the Euclidean norm.

The proof of the above lemma is easy and elementary, and hence we omit it here. Next, consider the product of two polytopes. Given d-dimensional polytopes $\mathcal{P}, \mathcal{Q}$ the product $\mathcal{P} \times \mathcal{Q}$ is defined as the set $\{(x, y)|x \in \mathcal{P}, y \in \mathcal{Q}\}$. The facet defining inequalities of the product of P, Q can be computed easily from the inequalities defining P and Q.

$$\begin{aligned} P &= \{x|A_1x \leq b_1\}, \\ Q &= \{y|A_2y \leq b_2\} \\ \Rightarrow P \times Q &= \{(x,y)|A_1x \leq b_1, A_2y \leq b_2\}, \end{aligned}$$

where $A_1 \in \mathbb{R}^{m_1 \times d_1}, A_2 \in \mathbb{R}^{m_2 \times d_2}, x \in \mathbb{R}^{d_1}, y \in \mathbb{R}^{d_2}, b_1 \in \mathbb{R}^{m_1 \times 1}, b_2 \in \mathbb{R}^{m_2 \times 1}$.

It is easy to see that the number of vertices of $\mathcal{P} \times \mathcal{Q}$ is the product of the number of vertices of $\mathcal{P}$ and that of $\mathcal{Q}$, and the number of facets of $\mathcal{P} \times \mathcal{Q}$ is the sum of the number of facets of $\mathcal{P}$ and that of $\mathcal{Q}$. Moreover, the dimension of $\mathcal{P} \times \mathcal{Q}$ is the sum of the dimensions of $\mathcal{P}$ and that of $\mathcal{Q}$.

Observation 1. *If c is the centroid of a polytope P then (c, c) is the centroid of $P \times P$.*

Suppose we are given an algorithm for finding ϵ-approximation of an arbitrary polytope contained in the unit hypercube. For example, for the simple algorithm that returns an arbitrary point inside the polytope, the approximation guarantee is $\frac{\sqrt{d}}{2}$. We consider similar algorithms whose approximation guarantee is a function of the ambient dimension of the polytope. Now suppose that for the given algorithm the approximation guarantee is $f(d)$. For some parameter k consider the k-fold product of P with itself $\overbrace{P \times \cdots \times P}^{k \text{ times}}$, denoted by P^k. Using the given algorithm one can find the $f(2^k d)$ approximation of P^k and using Lemma 4.3.3 one can then find the $\frac{f(2^k d)}{\sqrt{2}^k}$-approximation of P. This gives us the following bootstrapping theorem:

Theorem 4.3.4. *Suppose we are given an algorithm that computes a $\frac{\sqrt{d}}{g(d)}$-approximation for any polytope contained in the unit hypercube in polynomial time, where $g(.)$ is an unbounded monotonically increasing function. Then, one can compute an ϵ-approximation in time polynomial in the size of the polytope and $g^{-1}(\frac{\sqrt{d}}{\epsilon})$.*

In particular, if we have an algorithm with $d^{\frac{1}{2}-\delta}$ approximation guarantee for finding the centroid of any polytope for some fixed constant $\delta > 0$, then this algorithm can be used to construct a fully polynomial approximation scheme for the general problem.

4.4 Approximating centroid of unbounded polyhedra

The reader should note that the analysis of sections 4.2 and 4.3 remains valid even for the unbounded case (polyhedra). Even though we do not have any idea about the complexity of approximating the centroid of a polytope, now we show that for an arbitrary unbounded polyhedron the vertex centroid can not be $d^{\frac{1}{2}-\delta}$-approximated for any fixed constant $\delta > 0$. To show this we first prove that for an $\mathcal{H}$-polyhedron $P \subset \mathbb{R}^d$ the vertex centroid of P can not be $\frac{1}{d}$-approximated in polynomial time unless $P = NP$. This together with Theorem 4.3.4 completes the proof for hardness of $d^{\frac{1}{2}-\delta}$-approximation of the centroid of an $\mathcal{H}$-polyhedron.

The proof sketch is as follows: Given a boolean CNF formula ϕ, construct a graph $G(\phi)$ such that $G(\phi)$ has a "long" negative cycle if and only if ϕ is satisfiable. For a given graph G we define a polyhedron $P(G)$ such that every negative cycle in G is a vertex of $P(G)$ and vice-versa. From the properties of the vertex centroid of this class of polyhedron, we then prove that for any formula ϕ, $\frac{1}{d}$-approximating the vertex centroid of $P(G(\phi))$ would reveal whether ϕ is satisfiable or not.

For the above, we can use the constructions described in Chapter 3. We rephrase some crucial results from Chapter 3 as the following lemma:

Lemma 4.4.1. *For any 3-CNF ϕ with m clauses we can obtain a weighted graph $G = (V, E)$, $w : E \to \mathbb{R}$ and a polyhedron $P(G, w)$ with following properties:*

1. *G has $18m + 1$ edges and $3m$ short negative cycles. (Lemma 3.4.1)*

2. *Every negative cycle in G has weight -1. (Lemma 3.4.1)*

3. *There is an edge e that is in every long cycle but is not in any short cycle. (Remark 3.4)*

4. *Vertices of $P(G, w)$ are in one-to-one correspondence with the negative cycles of G. (Theorem 3.3.1)*

5. *Vertices of $P(G, w)$ have only $0/1$ coordinates. In fact, the coordinates of any vertex of $P(G, w)$ treated as a vector is just the characteristic vector of the corresponding negative cycle in G. (Theorem 3.3.1, Lemma 3.4.1)*

It is clear that the polyhedron $P(G, w)$ has dimension $18m + 1$ and $3m$ trivial vertices corresponding to the short negative cycles of G. Recall that checking whether $P(G, w)$ has more than the $3m$ vertices is NP-complete.

Since there is an edge e contained in all long cycles but not in any short cycle, the vertex centroid of $P(G, w)$ has value 0 in the coordinate corresponding to the edge e if there are no long negative cycles. For simplicity, we will refer to this coordinate axis as x_e. On the other hand, if there are $K \geq 1$ long negative cycles in G then in the centroid $x_e = \frac{K}{K+3m} \geq \frac{1}{3m+1}$. This implies that having an ϵ-approximation for the centroid of $P(G, w)$ for $\epsilon < \frac{1}{2(3m+1)}$ would reveal whether or not $P(G, w)$ has a non-trivial vertex and hence whether or not G has a long negative cycle. Thus we have the following theorem:

Theorem 4.4.2. *There is no polynomial algorithm that computes a $\frac{1}{d}$-approximation of the vertex centroid of an arbitrary $\mathcal{H}$-polyhedron $P \subset \mathbb{R}^d$, unless $P = NP$.*

An immediate consequence of Theorem 4.3.4 and Theorem 4.4.2 is that there is no polynomial algorithm that computes any "sufficiently non-trivial" approximation of the vertex centroid of an arbitrary $\mathcal{H}$-polyhedron unless $P = NP$. More formally,

Corollary 4.4.3. *There is no polynomial algorithm that $d^{\frac{1}{2}-\delta}$-approximates the centroid of an arbitrary d-dimensional $\mathcal{H}$-polyhedron for any fixed constant $\delta > 0$ unless $P = NP$.*

4.5 Open Problems

Although we can show that for unbounded polyhedra almost any non-trivial approximation of the vertex centroid is hard, we can not make a similar statement for the bounded case (*i.e. polytopes.* One interesting variant of Theorem 4.2.2 would be to consider a ball of radius r instead of a halfspace. If containment of vertex centroid in a ball of radius r can be decided in time polynomial in the number of inequalities defining the polytope, the dimension and r then one can perform a sort of random walk inside the polytope and approximate the centroid in polynomial time. We leave out the details of this random walk since we do not have a method to check containment inside a ball.

Chapter 5

Minkowski Addition and Related Operations

In this chapter, we study three fundamental operations on polytopes and provide hardness results for them. The results in this chapter have appeared in the proceedings of the 23rd annual Symposium on Computational Geometry 2007 [Tiw07]. A journal version has also appeared in Discrete and Computational Geometry [Tiw08].

5.1 Introduction

For polytopes P, Q in $\mathbb{R}^d$, the Minkowski addition $P+Q$, the convex hull of the union $CH(P \cup Q)$ and the intersection $P \cap Q$ are defined as:

$$\begin{aligned} P+Q &= \{x+y \mid x \in P, y \in Q\} \\ CH(P \cup Q) &= \{\lambda x + (1-\lambda)y \mid x \in P, y \in Q, 0 \leq \lambda \leq 1\} \\ P \cap Q &= \{x \mid x \in P \, and \, x \in Q\} \end{aligned}$$

We are interested in the complexity of performing these operations and providing non-redundant description of the resulting polytopes in appropriate representation. Since the worst case size of the output for all the three operations can be exponential in the size of input (see [FW05b, GS93]), it is natural to talk of *output sensitive* algorithms.

It is easy to see that computing the non-redundant $\mathcal{V}$-representation of $P+Q$

is easy if P and Q are $\mathcal{V}$-polytopes since the vertices of the result correspond to pairs of vertices from the operands and redundancy in the output can be removed by solving a polynomial number of Linear Programs. Computing the vertices of $CH(P \cup Q)$ from the vertices of P and the vertices of Q just amounts to removing redundancies. Similarly, computing the $\mathcal{H}$-representation of $P \cap Q$ is easy if P, Q are $\mathcal{H}$-polytopes using an LP solver for removing redundancies. Therefore, we are interested in other versions of these problems.

In this chapter, for the Minkowski sum, we consider and prove hardness results for the version where P, Q are $\mathcal{H}$-polytopes or, where one is a $\mathcal{V}$-polytope while the other is a polyhedral cone given by its facets. In both cases we want to compute the facets of the Minkowski sum $P+Q$. For $P \cap Q$, we consider and prove hardness results for the following three variants:

- P, Q each a $\mathcal{V}$-polytope, output the vertices of $P \cap Q$.
- P an $\mathcal{H}$-polytope, Q a $\mathcal{V}$-polytope, output the vertices of $P \cap Q$.
- P an $\mathcal{H}$-polytope, Q a $\mathcal{V}$-polytope, output the facets of $P \cap Q$.

The convex hull of the union; and intersection operations are related via polar duality. More precisely, if P, Q are two full dimensional polytopes (or polyhedra) in $\mathbb{R}^d$ both containing origin in the relative interior, then $P \cap Q$ is the polar dual of $CH(P^* \cup Q^*)$. The Minkowski sum, as we will shortly see, similarly relates to the convex hull of union via the so called Cayley trick.

5.1.1 The Cayley trick

The Cayley trick ([HRS00, Str94]) is a simple embedding of d-dimensional polytopes $P_1, \cdots, P_k$ into $\mathbb{R}^{d+k-1}$. The embedding is obtained by appending $\mathbf{e}_{i-1}$ to every point in P_i, where $\mathbf{e}_0 = \mathbf{0}$ and $\mathbf{e}_i$ is the i-th unit vector of $\mathbb{R}^{k-1}$, and taking the convex hull of all the embedded copies. It is easy to see that the Minkowski sum of these polytopes (up to a scaling) can be obtained from the Cayley embedding by intersecting the polytope obtained after the embedding with a suitable d-flat. To illustrate this, consider the case when $k = 2$. The Cayley embedding is obtained by putting a copy of P_1 in the hyperplane $x_{d+1} = 0$ and a copy of P_2 in the hyperplane $x_{d+1} = 1$ and taking the convex hull of both embedded polytopes. The Minkowski sum (scaled by a factor half), then, is the intersection of the resulting polytope with the hyperplane $x_{d+1} = \frac{1}{2}$.

The rest of the chapter is organized as follows. In the next section, we describe prior work related to performing these operations in appropriate representations and in Section 5.3 we describe the hardness results for computing the Minkowski sum of two polytopes. In Section 5.4 we establish hardness results for computing the intersection of two polytopes in various representations.

5.2 Related Work

The problem of enumerating the facets of $CH(P_1 \cup P_2)$, when both P_1 and P_2 are given by their facets, has been studied in [Bal88] and [FLL01]. Balas [Bal88] constructs polynomial algorithm for a special class of polytopes arising in 0-1 Mixed Integer programming, while Fukuda, Liebling and Lütolf [FLL01] present an algorithm that has polynomial complexity if the input polytopes satisfy certain general position assumptions. It is not clear if arbitrary polytopes can be made to satisfy the general position assumption as described in [FLL01]. The NP-hardness of computing the convex hull of the union of polytopes, as proved in this chapter, suggests that these assumptions are probably unrealistic for general polytopes.

Minkowski sums have been studied much more compared to the convex hull of the union. They frequently come up in computational algebra [GS93], robotics and motion planning, geometric convexity, computer graphics and many other areas. Gritzmann and Sturmfels [GS93] studied Minkowski sum in the context of computational algebra and gave (exponential) bounds on the number of faces of the Minkowski sum. They also gave examples of cases where the bounds are tight.

As noted before, exponential lower bounds suggest that we should search for output sensitive algorithms so that cases where the output is far from worst case can be handled efficiently. Fukuda and Weibel [Fuk04, FW05a] propose polynomial algorithms for enumerating all faces of the Minkowski sum of k polytopes where each polytope is given by its vertices. They do not consider the case when the input polytopes are described by facets and the facets of Minkowski sum are to be enumerated, and note this version of the problem to be open. Fukuda and Weibel, in another work (see [FW05b]), study Minkowski sums of special polytopes that are "well centered" and also provide better bounds on the number of faces for this special case.

Our main results state that the following decision problems are NP-complete and thus there is no output-sensitive algorithm for the corresponding enumeration problems unless $P = NP$:

- Given an $\mathcal{H}$-polytope P_1, an $\mathcal{H}$-polytope P_2, and an $\mathcal{H}$-polytope Q, is $P_1 + P_2 \neq Q$?
- Given an $\mathcal{H}$-cone P_1, a $\mathcal{V}$-polytope P_2, and an $\mathcal{H}$-polyhedron Q, is $P_1 + P_2 \neq Q$?
- Given a $\mathcal{V}$-polytope P_1, a $\mathcal{V}$-polytope P_2, and a $\mathcal{V}$-polytope Q, is $P_1 \cap P_2 \neq Q$?
- Given an $\mathcal{H}$-polytope P_1, a $\mathcal{V}$-polytope P_2, and a $\mathcal{V}$-polytope Q, is $P_1 \cap P_2 \neq Q$?
- Given an $\mathcal{H}$-polytope P_1, a $\mathcal{V}$-polytope P_2, and an $\mathcal{H}$-polytope Q, is $P_1 \cap P_2 \neq Q$?

For the first decision problem we provide a Turing reduction from another NP-complete problem. Usually reductions for proving NP-completeness employ Karp reduction. A problem $\mathcal{A}$ is said to be polynomial-time Turing reducible to problem $\mathcal{B}$ if one can construct a polynomial time algorithm for problem $\mathcal{A}$ using an oracle for $\mathcal{B}$. The more common Karp reduction allows only one call to the oracle and the answer from the oracle is returned, without modification, as the answer to the original decision problem A. For all other decision problems we provide the standard Karp reduction from some other NP-complete problem.

5.3 Hardness of Minkowski Addition

In this section we establish two hardness results about computing the facets of the Minkowski sum of two polytopes. We begin by proving the hardness of enumerating the facets of the Minkowski sum of two $\mathcal{H}$-polytopes. Consider the following decision version of the enumeration problem:

Problem IncompleteMinkowski
Input: $\mathcal{H}$-Polytopes P_1, P_2, Q.
Output: Yes, if $Q \neq P_1 + P_2$. No, otherwise.

Theorem 5.3.1. IncompleteMinkowski *is NP-complete.*

It was shown by Khachiyan *et. al* [KBB$^+$06] that it is NP-Hard to enumerate all vertices of a polyhedron given by its facets. The following theorem restates the result of [KBB$^+$06].

Theorem 5.3.2. *Given a polyhedron P in $\mathcal{H}$-representation and a set V of vertices of P, it is NP-complete to determine if P has some vertex* not *in V.*

Now, we prove that if we have an algorithm for deciding IncompleteMinkowski for arbitrary input polytopes, then we can invoke this oracle a polynomial number of times and decide for some set of vertices V and an $\mathcal{H}$-polyhedron P, whether $V \subsetneq vert(P)$.

Let $P = \{x|Ax \leq b\}$ be a polyhedron in $\mathbb{R}^d$ and $V \subseteq vert(P)$ with $|V| = n$. We want to determine whether $V \subsetneq vert(P)$ using a polynomial number of calls to an oracle for IncompleteMinkowski. For this we pick some direction and order the vertices of V in that direction. We assume that this direction is aligned with the x_d coordinate axis (after possibly applying a suitable affine transform). That is, if $\mathbf{e}_d$ is the unit vector $(0, \cdots, 0, 1)$ in $\mathbb{R}^d$ and $\mathbf{e}_d$ is thought to be the *upward* direction, then the vertices are considered in the order of increasing height, *i.e.* v_1 is the lowest vertex and v_n is the highest vertex. We also assume that any horizontal slice of P *i.e.* $P \cap \{x|x_d = c\}$ is a bounded polytope for any $c \in \mathbb{R}$. We justify this assumption later.

Now, consider vertices v_i and v_{i+1} for some fixed vertex subscript i and define three polytopes in the following way:

$$\begin{aligned} P_{-1} &= P \cap \{x_d = v_i \cdot \mathbf{e}_d\} \\ P_1 &= P \cap \{x_d = v_{i+1} \cdot \mathbf{e}_d\} \\ P_0 &= P \cap \left\{ x_d = \frac{v_i \cdot \mathbf{e}_d + v_{i+1} \cdot \mathbf{e}_d}{2} \right\} \end{aligned}$$

where the dot product $v_i \cdot \mathbf{e}_d$ is nothing but the x_d-coordinate (height) of v_i. Informally speaking, we are interested in three slices of the polyhedron: a top slice at the height of v_{i+1}, a bottom slice at the height of v_i and a slice at an intermediate height.

We claim that the middle slice is the Minkowski sum of the top and the bottom slices (with a scaling of half) if and only if there is no other (missing) vertex of P lying at an intermediate height between v_i and v_{i+1}. The following lemma states this formally.

Lemma 5.3.3. *$2P_0 \neq P_{-1} + P_1$ if and only if there exists some $v \in vert(P)$ that is not in V and $v_i \cdot \mathbf{e}_d < v \cdot \mathbf{e}_d < v_{i+1}.\mathbf{e}_d$.*

Proof. We prove the non-trivial direction only. Suppose some vertex $v \in vert(P)$ is not in V and $v_i \cdot \mathbf{e}_d < v \cdot \mathbf{e}_d < v_{i+1}.\mathbf{e}_d$ for the vertex subscript i under consideration. Without loss of generality we can assume that v lies above the hyperplane containing P_0. If so, there is an $u \in vert(P_{-1})$ such that $\overrightarrow{uv}$ lies on some edge of P. Clearly, $\overrightarrow{uv}$ intersects P_0, say at w. We claim that $2w \notin P_{-1} + P_1$.

Assume for the sake of contradiction that $2w \in P_{-1} + P_1$. Then there are $x \in P_{-1}$ and $y \in P_1$ such that $2w = x + y$. Since, any point on an edge of a polytope can be *uniquely* represented as the convex combination of the vertices defining the edge, it follows that $x = u$ and y is a vertex of P_1. This implies that v is a convex combination of x, y as well and hence, v can not be a vertex of P, a contradiction. □

To complete our algorithm for determining whether a given set V of vertices of an $\mathcal{H}$-polyhedron P is the complete vertex set of P, we also need to check the region below the lowest known vertex and above the highest known vertex. Thus, to complete the proof of Theorem 5.3.1 we need to be able to pick the direction "up" satisfying the following requirements:

(i) Every slice of P orthogonal to this direction is a polytope, *i.e* it is bounded.

(ii) Vertices of P have a unique ordering according to their heights in the "upward" direction. In other words, no two vertices of P have the same height.

(iii) We can find an upper bound on the height of all vertices of the polyhedron P. Furthermore, we require that this height be represented using a number of bits that is polynomial in the size of the input.

Note that for a direction satisfying requirements (i) and (ii), we can assume that the lowest vertex in the known set of V is also the lowest vertex of the polyhedron P. Also such a vertex can be found by solving a linear program and if V does not contain this vertex then clearly $V \subsetneqq vert(P)$. To check the region above the highest known vertex v of P, we take a slice of P at the height of v and another at the height defined by (iii), above which no vertex can possibly lie, and apply Lemma 5.3.3. We would also like that the direction satisfying the above requirements be represented using number of bits q that is polynomial in the number of bits used in the description of P and V.

5.3.1 Finding the sweep direction

To satisfy the first requirement, recall that a pointed polyhedron has a unique minimal representation as the Minkowski sum of a polytope and a cone. Also, the cone of the polyhedron $P = \{x|Ax \leq 1\}$ is just $cone(P) = \{x|Ax \leq 0\}$ with some inequalities possibly redundant. So a vector α such that $cone(P) \cap \{\alpha \cdot x \leq 1\}$ is bounded, satisfies the first requirement. Any vector picked from the interior $cone(A)$ does the trick, where every row of A is interpreted as a vector in $\mathbb{R}^d$. In particular the average of the row vectors of A satisfies requirement (i) and requires a number of bits that is polynomial in the size of A.

Let N be the set of the facet normals of $cone(A)$. Computing N is not an easy task but for our purposes we only need an upper bound on the size of the coefficients of these facet normals. It is known that the number of bits required to represent N is a polynomial in the number of bits required to represent A (See [GLS93] Page 164, Lemma 6.2.4). An immediate consequence of the same Lemma is that for any polyhedron P, a polynomial upper bound on the height of the topmost vertex can be computed. Thus assumption (iii) can be satisfied as long as the sweep direction needs a polynomial number of bits in its representation.

The first two conditions, for any possible sweep direction a, can be rewritten as:

$$\langle a, \eta \rangle < 0, \qquad \forall \eta \in N \tag{5.1}$$

$$\langle a, u - v \rangle \neq 0, \quad \forall u, v \in vert(P), u \neq v \tag{5.2}$$

where $\langle x, y \rangle$ is the inner product of the vectors x and y.

We already have a direction α satisfying (5.1). Now, consider the directions

$$\begin{aligned} \beta &= \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{d-1} \end{pmatrix} \\ \gamma &= x^d \alpha + \beta \end{aligned}$$

We show that for large enough x, the direction γ satisfies both the requirements (5.1) and (5.2), and that the number of bits needed for x, and hence for γ, is a polynomial in the size of the polyhedron P. We want that for γ

$$\begin{aligned}
\langle \gamma, \eta \rangle &= x^d \langle \alpha, \eta \rangle + \langle \beta, \eta \rangle < 0, \quad \forall \eta \in N && (5.3)\\
\langle \gamma, u - v \rangle &= x^d \langle \alpha, u - v \rangle + \langle \beta, u - v \rangle \neq 0, \quad \forall u, v \in vert(P), u \neq v && (5.4)
\end{aligned}$$

Notice that equations 5.3 and 5.4 involve polynomials in x. For large enough x the sign of the polynomial in equation 5.3 is the same as the sign of $\langle \alpha, \eta \rangle$. Since α satisfies (5.1), γ satisfies (5.1) as well for large enough x. It is also clear that the size of such an x need only be a polynomial in the size of the coefficients of the polynomial in 5.3.

Also, any polynomial in x evaluates to a non-zero value if x is larger than the largest possible root of the polynomial. Since the largest root of a polynomial has size polynomial in the size of its coefficients (See [Yap00], page 148, Lemma 6.7), the size of x required to satisfy equation 5.4 and hence condition (5.2) is a polynomial in the size of the coefficients involved in 5.4. This proves that we can pick a direction γ satisfying all the necessary conditions and requiring a number of bits that is polynomial in the size of P.

Thus, the polyhedron P and the vertex list V can be preprocessed so that their sizes remain polynomial, and so that if $V \subsetneq vert(P)$ then Lemma 5.3.3 can be used to find a missing vertex by checking the space between v_i and v_{i+1} for each i, and checking the space above the highest vertex. As stated before, any vertex of P requires a number of bits that is bounded by a polynomial in the size of P and so we can check the region above the highest vertex as well.

The above reduction proves that IncompleteMinkowski is NP-hard. To prove that IncompleteMinkowski is in NP as well, notice that given a hyperplane $h : \{a \cdot x = 1\}$ one can easily check whether it defines a facet of the Minkowski sum $P_1 + P_2$. To see this, let us consider the Cayley embedding of the two polytopes. If this hyperplane defines a facet of the Minkowski sum $P_1 + P_2$, then in the Cayley embedding as well, it corresponds to a facet of the convex hull of the union of the two polytopes. Given a hyperplane h one can find the faces of P_1 and P_2 that (possibly) define the corresponding facet of the Cayley embedding. For each P_i this can be done by simply translating the hyperplane away from the origin until it becomes a supporting hyperplane for P_i and taking the face of P_i contained in the hyperplane at this point. Whether these two faces define a facet of the Cayley embedding or not can be checked by just checking the dimension of the convex hull of the union of these two faces. This completes the proof of Theorem 5.3.1.

Following is an immediate corollary of Theorem 5.3.1:

Corollary 5.3.4. *Given two $\mathcal{H}$-polytopes $P_1, P_2 \in \mathbb{R}^d$, there is no output-sensitive algorithm that enumerates the facets of $P_1 + P_2$ unless $P = NP$.*

Since for an $\mathcal{H}$-polyhedron P and a subset of its vertices V, it is NP-complete to decide whether $V \subsetneq vert(P)$, we also have the following theorem:

Theorem 5.3.5. *Given an $\mathcal{H}$-cone P_1, a $\mathcal{V}$-polytope P_2, and an $\mathcal{H}$-polyhedron Q, it is NP-complete to determine whether $P_1 + P_2 \neq Q$.*

Proof. Any pointed polyhedron P has a unique minimal representation as the Minkowski sum of the polytope defined by its vertices and the cone of its extreme rays. Also, the cone of the extreme rays of the polyhedron $P = \{x|Ax \leq b\}$ is just $cone(P) = \{x|Ax \leq 0\}$. Redundant inequalities of $cone(P)$ can be removed using linear programming and hence $conv(V) + cone(P) = P$ if and only if $V = vert(P)$. Thus, an algorithm for enumerating the facets of the Minkowski sum of a $\mathcal{V}$-polytope and an $\mathcal{H}$-cone can be used to determine whether a given list of vertices of a polyhedron P is complete or not.

As noted earlier, given a hyperplane it can be easily checked whether it defines a facet of the Minkowski sum of P_1 and P_2. Thus, if $P_1 + P_2 \neq Q$ then there is a facet defining hyperplane for $P_1 + P_2$ that does not define a facet of Q. This proves that this decision problem is in NP as well. $\square$

It should be remarked that if the cone P_1 in Theorem 5.3.5 is represented by it's extreme rays then the problem of determining whether $P_1 + P_2 = Q$ or not, is equivalent to the problem of computing the V-representation of a polytope from the H-representation and vice-versa. The complexity status of the representation conversion problem remains open despite years of research [ABS97].

5.4 Hardness of Computing Intersection

Recall that the Minkowski sum of two polytopes can be computed via computing the convex hull of two polytopes using the Cayley embedding. To compute the Minkowski sum of polytopes P and Q in $\mathbb{R}^d$, we embed the polytopes in $\mathbb{R}^{d+1}$ by putting a copy of P in the hyperplane defined by $\{x_{d+1} = -1\}$ and a copy of Q

in the parallel hyperplane $\{x_{d+1} = 1\}$. If P_{-1} and Q_1 are the copies of P and Q respectively, then $P + Q$ is obtained (upto a scaling factor $\frac{1}{2}$) by taking the convex hull $CH(P_{-1} \cup Q_1)$ and intersecting it with the hyperplane $\{x_{d+1} = 0\}$.

Note that, the operand polytopes P_{-1} and Q_1 here are not full dimensional *i.e.* even though they are embedded in $\mathbb{R}^{d+1}$, neither of them has dimension $d+1$. However, one can easily ensure that these polytopes are full dimensional and both contain the origin in their relative interiors. To do this, we pick a point p in the relative interior of $CH(P_{-1} \cup Q_1)$ and construct a pyramid with base P_{-1} and p as the apex. It is easy to see that this can be done in polynomial time. Now we can pick another point q in the relative interior of this pyramid and create a pyramid with Q_1 as the base and q as the apex. Since the convex hull of the union of these two pyramids is the same as that of P_{-1} and Q_1 and their intersection is a full dimensional polytope, we can move origin in this common region. This together with Theorem 5.3.5 gives us the following theorem:

Theorem 5.4.1. *Given $\mathcal{H}$-polytopes $P_1, P_2, Q \in \mathbb{R}^d$, it is NP-complete to decide whether $CH(P_1 \cup P_2) \neq Q$.*

This can be dualized since each of the polytopes is full dimensional and contains origin in the relative interior. By considering the polar duals of P_1, P_2 and Q, each of which is a full dimensional $\mathcal{V}$-polytope containing the origin in the relative interior, we have the following theorem:

Theorem 5.4.2. *Given $\mathcal{V}$-polytopes $P_1, P_2, Q \in \mathbb{R}^d$, it is NP-complete to decide whether $P_1 \cap P_2 \neq Q$.*

Now we prove that the problem of computing either the facets or the vertices of the intersection of two polytopes is hard for the case where one of the polytopes is given by $\mathcal{H}$-representation and the other by $\mathcal{V}$-representation.

Theorem 5.4.3. *Given an $\mathcal{H}$-polytope P_1, a $\mathcal{V}$-polytope P_2, and an $\mathcal{H}$-polytope Q, it is NP-complete to decide whether $P_1 \cap P_2 \neq Q$.*

Proof. It is known ([FO85]) that given an $\mathcal{H}$-polytope P_1 and a $\mathcal{V}$-polytope P_2, it is NP-complete to decide whether $P_1 \nsubseteq P_2$. Clearly, $P_1 \subseteq P_2$ if and only if $P_1 \cap P_2 = P_1$. This implies that checking whether a given list of facets completely defines the intersection of an $\mathcal{H}$-polytope and a $\mathcal{V}$-polytope, is NP-hard.

The problem is also in NP because for given polytopes P_1, P_2 and Q, if $P_1 \cap P_2 \neq Q$ then Q has a vertex that does not lie in the intersection $P_1 \cap P_2$. A point lies in $P_1 \cap P_2$ if and only if it satisfies all the facet inequalities of P_1 and can be represented as the convex combination of the vertices of P_2. Both the tests can be performed in polynomial time for rational polytopes. □

As it turns out computing the vertices of the intersection of an $\mathcal{H}$-polytope and a $\mathcal{V}$-polytope is hard as well.

We know from Theorem 5.3.5 that it is NP-complete to decide whether a given list of facets of the Minkowski sum of an $\mathcal{H}$-cone P_1 and a $\mathcal{V}$-polytope P_2 in $\mathbb{R}^d$, is complete or not. As stated in the beginning of this section, we can embed P_1 and P_2 in $\mathbb{R}^{d+1}$ in two parallel hyperplanes and the Minkowski sum $P_1 + P_2$ is the intersection of the convex hull of P_1 and P_2 with an appropriate hyperplane. Also, we can pick points p and q in the convex hull of P_1 and P_2 such that the pyramids P_1' and P_2' obtained from P_1 with apex p and P_2 with apex q are full dimensional and have a full dimensional intersection. Thus,

Theorem 5.4.4. *Given an $\mathcal{H}$-polyhedron P_1 and a $\mathcal{V}$-polytope P_2, it is NP-hard to compute the facets of the polyhedron $CH(P_1 \cup P_2)$.*

Consider the polar duals of P_1 and P_2. Since both P_1 and P_2 are full dimensional and contain the origin in their relative interiors, the polar dual of P_2 is bounded *i.e.* an $\mathcal{H}$-polytope, and the polar dual of P_1 is a $\mathcal{V}$-polytopes with vertices $A \cup \{0\}$ if P_1 is represented as $Ax \leq \mathbf{1}$. The vertices of $P_1^* \cap P_2^*$ are in one-to-one correspondence with the facets of $CH(P_1 \cup P_2)$ and so we have the following theorem:

Theorem 5.4.5. *Given an $\mathcal{H}$-polytope P_1, a $\mathcal{V}$-polytope P_2 and a $\mathcal{V}$-polytope Q, it is NP-complete to decide whether $P_1 \cap P_2 \neq Q$.*

Chapter 6

VE-completeness: Projection of Polytopes

6.1 Introduction

As stated before, the problem of enumerating the vertices of a polytope given by its facets has been studied for a long time by a number of researchers. Still the complexity status of Vertex Enumeration problem (VE), for general dimension and for polytopes that are neither simple nor simplicial, is unknown. It is neither known to be in P nor is it known to be NP-complete. The dual problem of computing $\mathcal{H}$-representation from $\mathcal{V}$-representation, known as Convex Hull problem (CH), is equivalent to VE modulo solving a Linear Program. Thus, for rational input these two problems - VE and CH - are polynomial time equivalent and a polynomial output-sensitive algorithm for one can be used to solve the other in output-sensitive polynomial time.

In previous chapters we considered some problems whose efficient algorithm could yield an efficient algorithm for the vertex enumeration problem. Unfortunately these problems turned out to be NP-hard. This appears to be a general phenomenon. Problems like polytope containment [FO85], VE for polyhedra (Chapter 3), intersection of polytopes (Chapter 5) that arise from relaxing the problem of vertex enumeration very slightly are NP-hard even though the NP-hardness of VE itself is unknown. This suggests that it might be useful to define a complexity class based on VE and identify problems that are equivalent to vertex enumeration. In this chapter we define such a class and give examples of problems related to computing projections of polytopes that are equivalent to vertex enumeration. In chapter 7 we consider another problem of this class.

In order to be able to talk about the equivalence of Vertex Enumeration and projection, we will define a complexity class based on Vertex Enumeration. Keeping in line with other notions of completeness, we call an enumeration problem Φ VE-complete if any output-sensitive polynomial algorithm for VE can be used to solve Φ in output-sensitive polynomial time and vice-versa. Similarly, we call a problem VE-easy if it can be solved in output-sensitive polynomial time using an oracle for VE and we call a problem VE-hard if an oracle for this problem can be used to solve VE in output-sensitive polynomial time.

6.2 The Projection Problem

Given a polytope $P \subset \mathbb{R}^d$ the orthogonal projection $\pi(P)$ of P, onto a k-dimensional subspace spanned by the first k coordinate directions, is obtained by dropping the last $d-k$ coordinates from every point of P. More formally, the projection $\pi : \mathbb{R}^d \to \mathbb{R}^k$ is a map such that

$$\pi(P) = \{x \in \mathbb{R}^k | \exists y \in \mathbb{R}^{(d-k)}, (x, y) \in P\}.$$

In general the projection need not be orthogonal and the projection directions can be an arbitrary orthogonal set of vectors not necessarily aligned with the coordinate axes. In such cases one can apply an affine transform to align the projection directions along the coordinate axes, changing the polytope P in the process to another polytope Q and then considering an orthogonal projection of Q.

In this chapter we consider the problem of computing the projection of a polytope. Apart from the relation of projection to the vertex enumeration problem, which we will establish in this chapter, another motivation for considering this problem arises from the fact that one frequently needs to perform this operation is many areas like Control Theory, Constaint Logic Programming Languages, Constraint Query Languages etc ([Imb93]). And although many hardness proofs in this chapter are rather simple, they don't appear anywhere in the published literature. Furthermore, many papers seem to indicate that at least in some areas like

Input\Output	$\mathcal{V}$	$\mathcal{H}$	$\mathcal{HV}$
$\mathcal{V}$	poly	VE-Complete	VE-complete
$\mathcal{H}$	NP-hard	NP-hard	VE-complete
$\mathcal{HV}$	poly	VE-complete	VE-complete

Table 6.1: Complexity of computing projection of a polytope onto an arbitrary subspace, for various input and output representations.

Input\Output	$\mathcal{V}$	$\mathcal{H}$	$\mathcal{HV}$
$\mathcal{V}$	poly	VE-complete	VE-complete
$\mathcal{H}$	VE-hard	poly	VE-complete
$\mathcal{HV}$	poly	poly	poly

Table 6.2: Complexity of computing projection of a polytope for non-degenerate projection directions, for various input and output representations.

control theory (See, for example, [JKM04]) the quest for a polynomial algorithm (even for the versions that we prove NP-hard) is still on.

The projection problem has many variants depending on the input representation of the polytope P and the desired output representation of $\pi(P)$. In this chapter, we prove that for arbitrary projection directions, most versions of this problem are either VE-complete or NP-hard. Variants that are neither of these admit trivial polynomial algorithms. For example computing the vertices of $\pi(P)$ when P is a $\mathcal{V}$-polytope can be done simply by projecting each vertex and removing redundancies using linear programming. On the other hand, computing the facets or the vertices of $\pi(P)$ when P is an $\mathcal{H}$-polytope is NP-hard (Section 6.4).

Our main results are summarized in Table 6.1 and Table 6.2. Table 6.1 summarizes the complexity of computing projection along arbitrary directions while Table 6.2 summarizes the complexity of computing projection along "non-degenerate" projection directions. We will define the notion of non-degenerate projection directions in Section 6.5. We also consider the cases where the polytopes are either presented in both $\mathcal{H}$- and $\mathcal{V}$-representations or for the projection we require the computation of both representations. We denote these cases by $\mathcal{HV}$ for brevity.

Our results about the hardness, and the equivalence of vertex enumeration and computing projection (in most forms) imply that in all forms where the projection can not be computed by a trivial algorithm, finding an output-sensitive polynomial algorithm will be a challenging task. Equivalently, an output-sensitive polynomial algorithm for vertex enumeration will have significant impact in many fields outside algorithmic polytope theory.

Since for bounded polytopes containing the origin in relative interior, projection is an operation dual to intersection with a linear subspace, all our results can be dualized by interchanging vertices and facets, and replacing projection with intersection.

The rest of the chapter is organized as follows. In the next section we briefly review some related work about computing projection of a polytope. Our result section is divided into two parts. In section 6.4 we present the results about the complexity of computing the projection of a polytope along arbitrary directions and in section 6.5 we describe the complexity results for the case when the (orthogonal) projection directions are in some sense non-degenerate.

6.3 Related Work

Perhaps the best known algorithm for computing the facets of the projection of an $\mathcal{H}$-polytope is the Fourier-Motzkin elimination discovered by Fourier in 1824 and then rediscovered by Motzkin in 1936. This method is essentially the equivalent of Gaussian elimination for equations and works by eliminating one variable at a time. Since eliminating one variable from a system of m inequalities can result in $\left\lfloor \frac{m^2}{4} \right\rfloor$ facets, the algorithm can have a terrible running time in bad cases where the intermediate polytopes have very large (exponential) number of facets but the final output has only a small number of facets.

Many improvements have been made over the original algorithm (See [Imb93] for a survey) but there is no algorithm that has an output-sensitive polynomial running time. The natural question then is whether one can find a shortcut around the intermediate projection steps in the Fourier-Motzkin elimination and obtain an output-sensitive polynomial algorithm. As we will see in section 6.4, the answer is no if $P \neq NP$. Thus the lack of any output-sensitive algorithm, for computing the facets of the projection of an $\mathcal{H}$-polytope, is somewhat natural because the problem turns out be NP-hard.

Farkas' Lemma provides a way of generating a cone, although with different dimension, whose extreme rays correspond to the facets of the projection of an

$\mathcal{H}$-polytope [Bal98], but unfortunately it does not yield a bijective mapping and many extreme rays of the resulting cone may correspond to redundant inequalities in the projection. Balas [Bal98] found a way to get rid of these redundancies and provided a way to construct, given an $\mathcal{H}$-polytope P, another polyhedral cone W in polynomial time whose projection W' yields a one-to-one correspondence between the extreme rays of W' and the facets of $\pi(P)$. Also, W has polynomially many facets compared to P.

Jones, Kerrigan and Maciejowski [JKM04] describe an algorithm that runs in output-sensitive polynomial time (linear) if the projection is a non-degenerate polytope and the size and dimension of the projection is fixed. In general the algorithm is output-sensitive polynomial when the projection directions are in some sense non-degenerate. Assuming the projection to be non-degenerate is not justified in general, and as we will see in the next section it is impossible to get rid of such simplifying assumptions unless $P = NP$. We also present an output-sensitive algorithm for non-degenerate projection directions. Our algorithm for computing the projection of a polytope for non-degenerate projection directions (Section 6.5), although discovered independently by the author, is essentially the same as that of Jones et. al. and we include it in this chapter for completeness.

We will denote the projection of an n-dimensional polytope P onto a given d-dimensional subspace as $\pi_d(P)$. We will mostly omit the subscript and simply refer to the projection as $\pi(P)$ and the projection subspace will depend on the context.

Formally, we are interested in the following problem: Given a polytope $P \in \mathbb{R}^n$ in $\mathcal{H}$-, $\mathcal{V}$- or $\mathcal{HV}$-representation and a set Γ of k orthogonal projection directions defining the projection space, we want to compute the non-redundant $\mathcal{H}$-, $\mathcal{V}$- or $\mathcal{HV}$-representation of $\pi(P)$.

6.4 Projection onto arbitrary subspaces

Depending on the input and output form, we have nine variants of the problem. It is obvious that if the vertices are part of the input and one wants to compute the vertices of the projection, then each vertex can be projected trivially and the vertices that become redundant in the projection can be identified by solving one Linear Program per vertex. Hence, we have the following:

Lemma 6.4.1. *Given a polytope $P \subset \mathbb{R}^n$ in $\mathcal{V}$- or $\mathcal{HV}$-representation and an arbitrary projection subspace, non-redundant $\mathcal{V}$-representation of $\pi(P)$ can be computed in polynomial time.*

Also, it is easy to see that every polytope can be represented as the projection of a suitable simplex[1]. Moreover, given a polytope P by its vertices one can compute in polynomial time the vertices of this simplex Δ and the projection subspace such that P is the projection of Δ. Since it is trivial to compute the facets of a simplex given its vertices, Vertex Enumeration can be solved in output-sensitive polynomial time using any algorithm that computes the $\mathcal{H}$- or $\mathcal{HV}$-representation of projection from $\mathcal{V}$- or $\mathcal{HV}$-representation of a polytope.

Clearly, one can also use any polynomial algorithm for Vertex Enumeration to compute the $\mathcal{H}$- or $\mathcal{HV}$-representation of the projection of any polytope given in $\mathcal{V}$- or $\mathcal{HV}$-representation in polynomial time. Hence, we have the following easy lemma:

Lemma 6.4.2. *Computing the $\mathcal{H}$- or $\mathcal{HV}$-representation of the projection of a polytope given in $\mathcal{V}$- or $\mathcal{HV}$-representation is VE-complete.*

In what follows, we assume the input polytope P is of the form

$$P(x, y) = \{x, y | Ax + By \leq 1\}$$

and we want to compute the projection

$$\pi(P) = \{x \in \mathbb{R}^d | (x, y) \in P \text{ for some } y\},$$

where $A \in \mathbb{Q}^{m \times d}, B \in \mathbb{Q}^{m \times k}, y \in \mathbb{R}^k$. We also assume P to be full-dimensional and to contain the origin in its relative interior. For rational polytopes this assumption is justified because one can always find a point in the interior of the polytope via Linear Programming and move the origin to this point. We will sometimes omit details like $A \in \mathbb{Q}^{m \times d}, B \in \mathbb{Q}^{m \times k}, x \in \mathbb{R}^d, y \in \mathbb{R}^k$ where it can be inferred from the context.

We are now left with the three cases where the input polytope is given by $\mathcal{H}$-representation. As we will see now, computing either the facets or the vertices of

[1]Assuming that the vertices are numbered 0 through $m-1$, simply append e_i to the i-th vertex for $0 \leq i \leq m-d-2$ and e_{m-d-1} to all other vertices, where e_0 is the zero vector and e_i is the i-th unit vector in $\mathbb{R}^{m-d-1}$.

the projection in this case in hard while computing both facets and vertices of the projection is equivalent to Vertex Enumeration. Consider the following decision version of the problem:
Input: Polytopes $P = \{x, y|Ax + By \leq 1\}$ and $Q = \{x|A'x \leq 1\}$
Output: YES if $Q \neq \pi(P)$, NO otherwise.

We will now prove that this decision problem is NP-complete thus proving the NP-hardness of the enumeration problem.

Theorem 6.4.3. *Given a polytope* $P = \{x, y|Ax + By \leq 1\}$ *and* $Q = \{x|A'x \leq 1\}$ *it is NP-complete to decide if* $Q \neq \pi(P)$*.*

Proof. It is easy to see that deciding whether a given set of hyperplanes do not completely define the projection of a higher dimensional polytope is in NP. So the only thing remaining is to show that it is NP-hard as well.

It is known ([FO85]) that Given an $\mathcal{H}$-polytope $P = \{x|Ax \leq 1\}$ and a $\mathcal{V}$-polytope $Q = CH(V)$, it is NP-complete to decide whether $P \nsubseteq Q$. Clearly, $P \subseteq Q$ if and only if $P \cap Q = P$. Now, $P \cap Q$ is the projection of the following $\mathcal{H}$-polytope

$$\begin{aligned} Ax &\leq 1 \\ x - \sum_{v \in V} \lambda_v \cdot v &= 0 \\ \sum_{v \in V} \lambda_v &= 1 \\ \lambda_v &\geq 0, \forall v \in V \end{aligned}$$

The variables λ ensure that we consider only those points in P that can be represented as a convex combination of vertices of Q. One can further, in polynomial time, get a full dimensional representation of $P \cap Q$ by eliminating the $d - 1$ equations. The resulting polytope is a full-dimensional polytope in $\mathbb{R}^{|V|-1}$.

Since we are interested in only the (vector) variable x, the projection of this polytope along the axes corresponding to the variables λ_v gives us the facets of $P \cap Q$ in the subspace of variables x. It follows that $P \cap Q = P$ if and only if the projection of the above defined polytope onto the subspace of variables x has the $\mathcal{H}$-representation same as that of P *i.e.* $Ax \leq 1$.

Thus any polynomial algorithm for deciding whether a given set of hyperplanes completely define the projection of some high dimensional $\mathcal{H}$-polytope, can be used to decide whether an $\mathcal{H}$-polytope is contained in a $\mathcal{V}$-polytope, which is an NP-complete problem. □

Balas([Bal98]) has shown that for a given $\mathcal{H}$-polytope (P) and a set of projection axes, one can compute the facets of another pointed polyhedral cone W and a set of projection axes such that the facets of $\pi(P)$ are in one-to-one correspondence with the extreme rays $\pi(W)$[2]. The number of facets of W is polynomial in the number of facets of P. It is not difficult to modify the construction in [Bal98] so that W is bounded *i.e.* a polytope and the vertices in the projection of W are in one-to-one correspondence with the facets of the projection of P. For completeness we state the result of Balas and describe the modification here.

Lemma 6.4.4. *Given an $\mathcal{H}$ polytope P and a set of projection directions, there exists a polyhedral cone W, not in the same Euclidean space, and another set of projection directions for W such that the facets of $\pi(P)$ are in one-to-one correspondence with the extreme rays of $\pi(W)$. Furthermore, W has polynomially many facets compared to P and the facets of W can be computed in polynomial time.*

We will use the notion of polar duality to prove that the cone W obtained from the construction of Balas can be turned into a bounded polytope. For a polyhedral cone W in $\mathbb{R}^n$ with facet inequalities $Ax \leq 0$ and extreme rays V, where A and V are matrices with each row a vector in $\mathbb{R}^n$. The polar dual W^* has the roles of the extreme rays and facets reversed. In particular, the facet inequalities of W^* are $Vx \leq 0$ and the extreme rays are the row vectors of A. We again refer the reader to [Grü03, Zie95] for more details of the properties of this duality.

Lemma 6.4.5. *Given a pointed polyhedral cone $W \in \mathbb{R}^n$ in either $\mathcal{H}$- or $\mathcal{V}$-representation, and a set of projection directions Γ one can construct, in polynomial time, a polytope $Q \in \mathbb{R}^n$, in the same representation as W, such that the extreme rays of $\pi(W)$ are in one-to-one correspondence with the vertices of $\pi(Q)$, where both the projections are onto the same subspace.*

Proof. Clearly, none of the projection directions lie in W otherwise the projection spans the whole subspace. Let W^* be the polar dual of W. For any vector α in the interior of W^* the hyperplane $\alpha \cdot x = 0$ touches W only at the origin and hence

[2]Note that the projections of P and W are defined in different spaces and should not be confused as the same projection map despite the abuse of notation here.

$W \cap \{x | \alpha \cdot x \leq 1\}$ is a bounded polytope. It is actually a pyramid with the origin as the apex.

Now consider the projection $\pi(W)$, which is a pointed cone with origin as apex. This cone is a full-dimensional cone in the subspace containing it and we can consider its polar dual in that subspace. Let $\pi^*(W)$ be the polar dual of $\pi(W)$. For any vector $\alpha^{'}$ in the interior of $\pi^*(W)$, $\pi(W) \cap \{x | \alpha^{'} \cdot x \leq 1\}$ is a bounded polytope. Moreover, for such an $\alpha^{'}$, $\gamma_i \cdot \alpha^{'} = 0$ for all $\gamma_i \in \Gamma$. Since $\pi(W)$ is a pointed cone such an $\alpha^{'}$ exists.

Since $\pi^*(W)$ can be obtained as the intersection of W^* with $\bigcap_{\gamma_i \in \Gamma} \{\gamma_i \cdot x = 0\}$, a vector $\alpha^{'}$ in the interior of $\pi^*(W)$ can be computed in polynomial time if one knows either the vertices or facets of W. Also, $\alpha^{'}$ lies in the interior of W^* as well. Define $Q = W \cap \{\alpha^{'} \cdot x \leq 1\}$. Given the extreme rays (facets respectively) of W and the projection directions Γ one can compute the vertices (facets respectively) of Q in polynomial time.

Since $\alpha^{'}$ is orthogonal to each of the projection directions, the vertices and facets of $\pi(Q)$ are in one-to-one correspondence with the extreme rays and the facets of $\pi(W)$. □

Theorem 6.4.3 together with Lemma 6.4.4 and Lemma 6.4.5 gives the following:

Theorem 6.4.6. *Given a polytope $P = \{x, y | Ax + By \leq 1\}$ and $Q = CH(V)$ it is NP-complete to decide if $Q \neq \pi(P)$.*

Now we consider the last variant of the projection problem where we are given an $\mathcal{H}$-polytope and we want to compute the $\mathcal{HV}$-representation of the projection. As it turns out, although computing either the vertices or facets of the projection is NP-hard, computing both vertices and facets is VE-complete.

Before we prove this, we would like to remark that the notion of output-sensitiveness can have various meanings. An output-sensitive polynomial algorithm for an enumeration problem (like VE) could enumerate vertices such that a new vertex is reported within incremental polynomial delay *i.e.* each new reporting takes time polynomial in the input and the output produced so far. It is equally conceivable that the algorithm takes total time polynomial in the input and output but there is no guarantee that successive reportings take only incremental polynomial delay. We will assume that if we have an output-sensitive algorithm of the latter kind, then we actually know the complexity of its running time. Under this assumption the two notions are same for VE.

To see why this is true, consider the following. Given an $\mathcal{H}$-polytope P and a $\mathcal{V}$-polytope Q, determining whether $P = Q$ is polynomial time equivalent to VE (See [ABS97]). Also, solving this problem gives an algorithm for VE that is not only output-sensitive polynomial but also has a polynomial delay guarantee. If we have an enumeration algorithm that has no guarantee of polynomial delay between successive outputs, but for which we know the running time, then we can use this procedure to create a polynomial algorithm for deciding the equivalence of $\mathcal{H}$- and $\mathcal{V}$-polytopes: Simply compute the time needed by the algorithm to enumerate all vertices of P assuming $P = Q$ and run the enumeration algorithm for the time required to output $|V| + 1$ vertices. If the procedure stops then we can compare the list of vertices of P with that of Q in polynomial time. If, on the other hand, the procedure doesn't finish within the given time then P must have more vertices than Q and hence, $P \neq Q$.

So for proving VE-completeness in the next theorem, when we assume the existence of an output-sensitive polynomial algorithm for VE, we also assume that this algorithm has a guarantee of polynomial delay between successive outputs. Although we will work with the Convex Hull problem which is the dual version of VE, with a slight abuse of language we will refer to this dual problem as VE as well.

Theorem 6.4.7. *Given a polytope $P = \{x, y | Ax + By \leq 1\}$ it is VE-complete to compute the facets and vertices of $\pi(P)$.*

Proof. Since every polytope $P \in \mathbb{R}^n$ given by m vertices can be converted to a $(m-1)$-dimensional simplex Δ such that P is a projection of Δ it is clear that computing $\mathcal{HV}$-representation of the projection of an $\mathcal{H}$-polytope is VE-hard. It is non-trivial to prove that the problem is VE-easy well. Recall that computing the vertices of P first, followed by projecting these vertices and computing the facets of the projection might be too expensive since P may have many more vertices than $\pi(P)$.

To prove that this problem is also VE-easy, we give an algorithm that uses a routine for VE to enumerate the facets and vertices of $\pi(P)$. The algorithm proceeds as follows: At any point we have a list of vertices V of $\pi(P)$ and we want to verify that V indeed contains all vertices of $\pi(P)$. If the list is not complete, we want to find another vertex of $\pi(P)$ that is not already in V. To do this, we start enumerating facets of $CH(V)$ and we verify that each generated facet is indeed a facet of $\pi(P)$. This is easy to check because of the following:

Suppose $h = \{x|a \cdot x = 1\}$ be a hyperplane in the projection space. We say that h intersects P properly if the intersection $P \cap \{(a, \overbrace{0, \cdots, 0}^{k\text{times}}) \cdot (x, y) = 1\}$ has dimension $n - 1$. We will call such an intersection a *proper* intersection.

We claim that the defining hyperplane of every facet f of $CH(V)$, that is not a facet of $\pi(P)$, intersects P properly. To see this, pick a point x_1 in the relative interior of f. Such a point exists because $CH(V) \subset \pi(P)$ if some facet f of $CH(V)$ is not a facet of $\pi(P)$. This point also lies in the relative interior of $\pi(P)$. Also, there is a point (x_1, y_1) that lies in the relative interior of P that projects to x_1. Clearly the hyperplane $\{(a, \overbrace{0, \cdots, 0}^{k\text{times}}) \cdot (x, y) = 1\}$ contains (x_1, y_1) and hence the hyperplane defining f intersects P properly.

It follows that, if V does not contain all vertices of $\pi(P)$ then there exists a facet $f = \{x|a \cdot x = 1\}$ of $CH(V)$ intersecting P properly. So if the enumeration procedure for facets of $CH(V)$ stops and none of the facets intersect P properly then V contains all the vertices of $\pi(P)$. If some intermediate facet $\{a \cdot x = 1\}$ of $CH(V)$ does intersect P properly then maximizing the objective function $(a, \overbrace{0, \cdots, 0}^{k\text{times}}) \cdot (x, y)$ over P produces a vertex of P that also gives a vertex v of $\pi(P)$ upon projection. Moreover this vertex is not in the list V. Thus, if V is not a complete vertex description of $\pi(P)$ we can find another vertex of $\pi(P)$ in polynomial time. This gives an output-sensitive polynomial algorithm for enumerating all facets and vertices of $\pi(P)$. Hence, computing all vertices and facets of the projection $\pi(P)$ of an $\mathcal{H}$-polytope P is VE-easy as well. $\square$

Given that it is widely believed that $P \neq NP$ and VE has been studied quite closely by a number of researchers, one can infer that computing the projection is going to be a challenging problem for arbitrary projection directions. Also, one should note that if the projection is known to be a simple of simplicial polytope then computing both facets and vertices of the projection can be done in output-sensitive polynomial time because VE for simple or simplicial polytopes can be done in output-sensitive polynomial time ([AF92, BFM98]). But if the projection yields a degenerate polytope then we do not have such an algorithm.

6.5 Projection along non-degenerate directions

Now we show that if the projection directions are in some sense non-degenerate, then this problem can be solved in output-sensitive polynomial time in many cases even if the projection is neither simple nor simplicial. To make this notion precise, note that if P is the input polytope then every face of projection $\pi(P)$ is the shadow of some proper face of P. Call the maximal dimensional face f' of P a *pre-image* of the face f of $\pi(P)$ if f is obtained by projecting all vertices defining f' and taking their convex hull. In general, the dimensions of f and f' are not the same. This can happen if some projection directions lie in the affine hull of f'. We call a set of projection directions *non-degenerate* with respect to P if no directions lie in the affine hull of any face of P.

Also, for non-degenerate projection directions a face f of $\pi(P)$ and its pre-image f' have the same affine dimension. This is easy to see because a projection reduces the dimension of some face f' of P if and only if the projection direction lies in the affine hull of f' which is not possible for non-degenerate projection directions. Thus,

Fact 2: For non-degenerate projection directions and a face f of $\pi(P)$, if f' is the pre-image of f then $dim(f) = dim(f')$.

Now, given a polytope P in $\mathcal{H}$-representation and a set of non-degenerate projection directions Γ we want to compute the facets of the projection $\pi(P)$. Again, we assume that the facets of P are presented as inequalities of the form $Ax \leq 1$, where A is a matrix of size $m \times (d + k)$ and the projection has dimension d. Since, we will need to solve Linear Programs we also assume that the polytope is rational *i.e.* the entries in A are rational numbers. We will omit Γ from the discussion below and assume that the projection directions are aligned along a subset of coordinate axes. If not, we can apply a suitable affine transform to P depending on the orthogonal projection directions. Thus the reader should bear in mind that the polytope in what follows is a result of an affine transformation determined by a set of non-degenerate orthogonal projection directions.

Our algorithm for enumerating the facets of $\pi(P)$ proceeds as follows: Given a partial list of facets of $\pi(P)$, for each facet f we identify its pre-image f' in P. For each of these faces of P we identify their $(d-2)$-dimensional faces and among all such $(d-2)$-faces of f' some give rise to ridges in $\pi(P)$. We identify which faces form the pre-image of some ridge of $\pi(P)$ and from the corresponding ridge, we identify the two facets defining this ridge, thus finding a new facet of $\pi(P)$ if the current list of facets is not complete.

Lemma 6.5.1. *Given an $\mathcal{H}$-polytope P and a facet f of its projection $\pi(P)$, one can find the facets of P defining the pre-image of f in polynomial time.*

Proof. Let $\{x \in \mathbb{R}^d | a \cdot x \leq 1, a \in \mathbb{Q}^d\}$ be the halfspace defining the facet f of $\pi(P)$. Clearly, the hyperplane h in $\mathbb{R}^{d+k}$ with normal $a' = (a, \overbrace{0, \cdots, 0}^{k \text{ times}})$ defines the supporting hyperplane $\{x \in \mathbb{R}^{d+k} | a' \cdot x = 1\}$. Also, $P \cap h$ is a face of P and is exactly the pre-image of f. A facet F of P contains this face iff $P \cap h \cap F$ has the same dimensions as $P \cap h$. Thus, one can find all the facets of P containing the pre-image of f in time polynomial in the size of A. □

The next lemma follows immediately from the non-degeneracy of the projection directions, so we mention it without the proof (See Fact 2).

Lemma 6.5.2. *Given P and a facet f of its projection $\pi(P)$, if g is another facet of $\pi(P)$ sharing a ridge with f then the pre-images f' and g' share a face in P. Furthermore,*

$$dim(f' \cap g') = dim(f \cap g) = dim(f') - 1 = dim(g') - 1 = d - 2$$

Since the facets of P are known, we can identify all $(d-2)$-faces of f'. The number of these faces is at most m for each pre-image f' and since f' is itself a polytope of dimension $d-1$, we can compute the non-redundant inequalities defining the facets ($d-2$-dimensional faces) of f'[3]. At this point, what remains is to identify these ridges and the facets defining these ridges. The following two lemmas achieve this.

Lemma 6.5.3. *Let $P = \{(x,y) | A \cdot (x,y) \leq 1\}$ be a polytope in $\mathbb{R}^d \times \mathbb{R}^k$, where $A \in \mathbb{Q}^{m \times (d+k)}$, $x \in \mathbb{R}^d, y \in \mathbb{R}^k$. Also, let f be a $(d-2)$-face of P defined as $f = \{(x,y) | A' \cdot (x,y) = 1, (x,y) \in P\}$, where $A' \subset A$. Then, f defines a ridge in the projection $Q(x)$ if and only if*

[3]Removing redundancies can be achieved via Linear Programming.

- *there exists $\alpha \in \mathbb{R}^d$ such that $(\alpha, \overbrace{0, \cdots, 0}^{k\ times}) \in conv(A^{'})$, where each row of $A^{'}$ is treated as a point in $\mathbb{R}^{d+k}$. And,*

- *The feasible region of all such α is a line segment.*

It is not difficult to see that this lemma is just a rephrasing of the basic properties of supporting hyperplanes of a polytope. In other words, any hyperplane whose normal is a convex combinations of the normals of facets defining the face f, is a supporting hyperplane of P and vice-versa. Furthermore, if the normal lies in the subspace where the projection $\pi(P)$ lives, then it is also a supporting hyperplane of $\pi(P)$. Also, the normals of all hyperplanes that support a polytope at some ridge, when treated as points, form a 1-dimensional polytope *i.e.* a line segment. This formulation allows us to check in polynomial time whether a $(d-2)$-face of P forms a pre-image of some ridge of $Q(x)$.

Lemma 6.5.4. *The end points of the feasible region of α in lemma 6.5.3 are the normals of the facets of $\pi(P)$ defining the ridge corresponding to face f.*

As noted before, the normals of the hyperplanes supporting a polytope at a ridge r form a line segment when viewed as points. The end points of the segment represent the normals of the two facets defining the ridge r. This lemma ensures that given a pre-image of some ridge of $\pi(P)$, one can compute the normals of the two facets of $\pi(P)$ defining the ridge r by solving a polynomial number of linear programs each of size polynomial in the size of input.

Putting everything together we get the following theorem:

Theorem 6.5.5. *Given an $\mathcal{H}$-polytope P and a set of non-degenerate orthogonal projection directions Γ one can enumerate all facets of $\pi(P)$ in output sensitive polynomial time.*

Note, that this also gives an output-sensitive polynomial algorithm for the case when the input is an $\mathcal{HV}$-polytope irrespective of the output form as long as the projection directions are non-degenerate. Also, if the vertices of P are given then some tests like those in Lemma 6.5.3 and 6.5.4 become easier. We leave the proof of this to the reader since they do not affect our main argument about an output-sensitive polynomial algorithm.

Corollary 6.5.6. *Given an $\mathcal{HV}$-polytope P and a set of projection directions Γ that are non-degenerate with respect to P there is an algorithm that can enumerate the vertices and/or facets of $\pi(P)$ in output-sensitive polynomial time.*

It is easy to see that computing the vertices of the projection of an $\mathcal{H}$-polytope along non-degenerate directions has VE as a special case[4]. Hence, enumerating vertices of the projection of an $\mathcal{H}$-polytope along non-degenerate projection directions remains VE-hard even though it is not clear if it remains NP-hard. Similarly one can argue that the complexity status of computing the projection of a $\mathcal{V}$-polytope along non-degenerate projection directions remains the same as that of computing the projection along arbitrary directions.

[4]Just pick the set of projection directions to be the empty set.

Chapter 7

Union of Cones

The results in this chapter are a joint work with Khaled Elbassioni and have appeared in the proceedings of 20th Canadian Conference on Computational Geometry [ET08].

7.1 Introduction

In the previous chapter we defined the class of problems equivalent to the vertex enumeration problem for polytopes and considered the problem of computing projection of polytopes that, in some versions of the problem, turns out to be VE-complete. In this chapter we consider another problem whose complexity can be related to that of vertex enumeration in many forms. We call this the cone covering problem, where one is given a set of polyhedral cones in $\mathbb{R}^d$ and one wants to know whether a given object is covered by these cones or not. Naturally we do not consider arbitrary objects that are to be covered but only very specific convex objects. We will specify these objects later.

Apart from its relation to vertex enumeration, our motivation for studying the above covering problems comes from the problem of checking whether the union of a given set of polytopes is convex or not. Bemporad, Fukuda and Torrisi [BFT01] gave polynomial-time algorithms for checking if the union of $k = 2$ polyhedra is convex, and if so finding this union, no matter whether they are given in $\mathcal{V}$ or $\mathcal{H}$ representations. They also gave necessary and sufficient conditions for the union of a finite number of convex polytopes in $\mathbb{R}^d$ to be convex, and asked whether these conditions can be used to design a polynomial time algorithm for checking if the union is convex. Bárańy and Fukuda give slightly stronger conditions in [BF05]. It will follow from our results that, if both d and k are part of the input, then these conditions can not be checked in polynomial time unless P=NP.

7.2 Problem description

In this chapter we are interested in the complexity of covering problems of the following form:

ConeCover($\mathcal{C}, D$): Given a collection of polyhedral cones $\mathcal{C} = C_1, \ldots, C_N$, and a convex set D, does $\bigcup_{i=1}^{N} C_i \not\supseteq D$?

Unless otherwise specified, all the cones considered throughout the paper will be assumed to be pointed, i.e., contain no lines, or equivalently, have only one vertex, namely the origin. As we will see, the complexity of the above problem depends on what the convex set D is, how the cones are represented, and whether they are disjoint or not. We consider 3 different factors, namely:

(f1) whether the cones in $\mathcal{C}$ are given in $\mathcal{V}$- or $\mathcal{H}$-representations, or both representations ($\mathcal{HV}$),

(f2) what the set D is: we consider $D = \mathbb{R}^d$ and $D = \mathbb{R}^k$ for some arbitrary $k \leq d$.

(f3) whether the cones in $\mathcal{C}$ are

- (f3)-(I): pairwise disjoint in the interior and intersect only in faces;
- (f3)-(II): pairwise disjoint in the interior, *i.e.* $\text{int}(C_i) \cap \text{int}(C_j) = \emptyset$ for all $i \neq j$, but can intersect anywhere on the boundaries; and
- (f3)-(III): not necessarily pairwise disjoint.

We denote by ConeCover[$F1, F2, F3$] the different variants of the problem, where $F1 \in \{\mathcal{V}, \mathcal{H}\}$, $F2 \in \{\mathbb{R}^k, \mathbb{R}^d\}$ and $F3 \in \{I, II, III\}$ describes cases (f1)-(I), (f2)-(II), and (f3)-(III). Our results on the complexities of these problems are summarized in Table 7.1.

	$\mathbb{R}^d$			$\mathbb{R}^k$		
	I	II	III	I	II	III
$\mathcal{V}$	VE-hard	VE-hard	NPC	NPC	NPC	NPC
$\mathcal{H}$	P	?	NPC	P	?	NPC
$\mathcal{HV}$	P	P	NPC	P	?	NPC

Table 7.1: Complexity of Cone Covering problem for various input representations.

Similar to the cone covering problem one can also consider the related problem of polytope covering:

PolytopeCover($\mathcal{P}, D$): Given a collection of polytopes $\mathcal{P} = P_1, \ldots, P_N$, and a convex polytope D, does $\bigcup_{i=1}^{N} P_i \not\supseteq D$?

In this chapter we also show that if all the polytopes in the above problem are given either by only vertices or only facets then the problems is NP-complete. As a consequence of this we prove that deciding whether the union of a given set of $\mathcal{H}$- or $\mathcal{V}$-polytopes is convex or not, is NP-complete. This answers a question of Bemporad, Fukuda and Torrisi [BFT01]

7.3 Covering sets by Cones

The following decision version of the vertex enumeration problem is known to be equivalent to the enumeration problem [ABS97].

Given an $\mathcal{H}$-polytope $P \subseteq \mathbb{R}^d$ and a subset of its vertices $V \subseteq \mathcal{V}(P)$, Is $P = \text{conv}(V)$?.

Let P be the polytope defined as $\{x \in \mathbb{R}^d | Ax \leq \mathbf{1}\}$, where $A \in \mathbb{R}^{m \times d}$. For any vertex v of P, consider the cone of all vectors c such that v is the solution of the following linear program:

$$\begin{aligned} &\max c^T \cdot x \\ s.t. \quad &Ax \leq 1 \end{aligned}$$

For every vertex v of P, this cone is uniquely defined. We call this cone the *maximizer cone* of v. Such a maximizer cone can be defined for every proper face of a polytope. The union of all such cones is also known as the *normal fan* of a polytope [Zie95]. It is easy to see that if A is the maximal subset of rows of A such that $A'v = 1$, then the maximizer cone of v is the conic hull of the rows of A' treated as vectors in $\mathbb{R}^d$.

Theorem 7.3.1. *Problem* ConeCover$(\mathcal{V}, \mathbb{R}^d, I)$ *is VE-hard.*

Proof. Given an $\mathcal{H}$-polytope P and a subset of its vertices V, the $\mathcal{V}$-representation of the maximizer cone for each vertex in V can be computed easily from the facets of P. Clearly, the union of these cones covers $\mathbb{R}^d$ if and only if $P = conv(V)$. To see this, note that if $P \neq conv(V)$ then P has a vertex v not in V and any vector in the relative interior of the maximizer cone of v does not lie in any of the cones corresponding to the given vertices. □

Theorem 7.3.2. *Problem* ConeCover$(\mathcal{V}, \mathbb{R}^k, I)$ *is NP-complete.*

Proof. ConeCover$(\mathcal{V}, \mathbb{R}^k, \mathrm{I})$ is clearly in NP. Now, given an $\mathcal{H}$-polytope $P \subset \mathbb{R}^d$, an affine subspace $\mathbb{R}^k$ and a $\mathcal{V}$-polytope $Q \subset \mathbb{R}^k$, it is NP-complete to decide whether Q is the projection of P onto the given subspace (Chapter 6). We give a polynomial reduction from this problem to ConeCover$(\mathcal{V}, \mathbb{R}^k, \mathrm{I})$.

Every vertex v of Q is an image of some (possibly more than one) vertices of P. Pick any such vertex and call it v'. We associate the maximizer cone of v' with v and refer to it as $\mathcal{C}(v)$. The $\mathcal{V}$-representation of the cone associated with each vertex of Q can be easily computed from the matrix A.

It is not difficult to see that if Q is not the projection of P onto the given subspace $\mathbb{R}^k$, then one can find a direction c parallel to the given subspace such that a vertex that maximizes $c^T x$ in P is such that its projection is a vertex of the projection of P but not of Q. This would imply that the unions of the cones $\mathcal{C}(v)$ for each vertex of Q does not cover $\mathbb{R}^k$. Conversely, if the union of $\mathcal{C}(v)$ for each vertex of Q does not cover $\mathbb{R}^k$ then some vertex of the projection of P is missing in Q. In particular any direction c that is parallel to the given projection subspace, but is not covered by any of the cones $\mathcal{C}(v)$ of vertices of Q, corresponds to a missing vertex of the projection. Hence, the union of cones $\mathcal{C}(v)$ for each vertex v of Q covers $\mathbb{R}^k$ if and only if Q is the projection of P. Also, all these cones intersect each other only at some proper face.

□

For a given set of $\mathcal{H}$-cones, if the union does not cover $\mathbb{R}^d$ then there exists a facet, with facet normal $a \in \mathbb{R}^d$, of at least one of these cones such that picking a point p in the interior of this facet, $p + \epsilon a$ lies outside every cone, for all $\epsilon > 0$. Lets call this facet a *witness facet*, and p a *witness point* of the fact that $\mathbb{R}^d$ is not covered.

Theorem 7.3.3. ConeCover$(\mathcal{H}, \mathbb{R}^d, I)$ *can be solved in polynomial time.*

Proof. If the cones are allowed to intersect only at common faces, then every point in the interior of a witness facet is a witness point. Thus, one can determine in polynomial time whether the union of the given cones cover $\mathbb{R}^d$ or not, by picking a point in the interior of every facet, with normal a, of every cone and using linear programming to check if $p + \epsilon a$ lies outside every cone for every $\epsilon > 0$. □

Theorem 7.3.4. ConeCover$(\mathcal{HV}, \mathbb{R}^d, II)$ *can be solved in polynomial time.*

Proof. It is easy to see that if the cones are allowed to intersect only on the boundary, and if the union of the given cones does not cover $\mathbb{R}^d$, then the extreme rays[1] of any (possibly non-convex) "hole" are also the extreme rays of some cone. For any such extreme ray w, if one considers a d-dimensional ball of radius ϵ centered at some point on w, then for small enough ϵ some part of this ball is not covered by any of the given cones.

Consider all the halfspaces $\{ax \leq 0\}$ corresponding to the facets of the input cones that contain w, *i.e.* $aw = 0$. Let A be the matrix with each row the normal vector of such a halfspace. The union of the given cones does not cover $\mathbb{R}^d$ if and only if $\{Ax \geq 0\}$ define a full dimensional region. This can be easily checked via linear programming.

□

Fact 1. *For any $k \in \mathbb{N}$, we can write $\mathbb{R}^k = \cup_{i=1}^{k+1} R_i$, where $R_1, \ldots, R_{k+1}$ are pointed cones, pairwise-disjoint in the interior, whose $\mathcal{H}$- and $\mathcal{V}$-representations can be found in in polynomial time.*

Let $C_1 = \{x \in \mathbb{R}^d \mid A_1 x \leq \mathbf{0}\} = \text{cone}\{S_1\}$ and $C_2 = \{x \in \mathbb{R}^d \mid A_2 x \leq \mathbf{0}\} = \text{cone}\{S_2\}$, where $A_1 \in \mathbb{R}^{l \times m}, A_2 \in \mathbb{R}^{r \times n}$ and $S_1 \subseteq \mathbb{R}^m, S_2 \subseteq \mathbb{R}^n$, be two polyhedral cones. The *cartesian product* of C_1 and C_2, is defined as:

[1]An extreme ray of a non-convex polyhedral cone C is defined as the rays that must occur as an extreme ray of at least one simplex cone in any subdivision of C using only simplex cones.

$$\begin{aligned} C_1 \times C_2 &= \{(x,y) \in \mathbb{R}^m \times \mathbb{R}^n |\ A_1 x \leq \mathbf{0},\ A_2 y \leq \mathbf{0}\} \\ &= \text{cone}\left(\left\{\begin{pmatrix} v \\ \mathbf{0} \end{pmatrix} : v \in S_1\right\} \bigcup \right. \\ &\qquad \left. \left\{\begin{pmatrix} \mathbf{0} \\ v \end{pmatrix} : v \in S_2\right\}\right) \end{aligned}$$

Theorem 7.3.5. *Problems* ConeCover($\mathcal{HV}, \mathbb{R}^d$, *III*) *is NP-complete.*

Proof. Clearly the problems is in NP since a direction exists outside the union of the given cones if they do not cover $\mathbb{R}^d$. Since the facet defining inequalities of each cone are known, it can be easily verified that such a given direction indeed lies outside each of the given cones. For proving its NP-hardness, we use a reduction from the following problem:

Sat($V, \mathcal{F}, \mathcal{G}$): Given a finite set V and two hypergraphs $\mathcal{F}, \mathcal{G} \subseteq 2^V$, is there a set $X \subseteq V$ such that:

$$X \not\supseteq F \text{ for all } F \in \mathcal{F} \text{ and } X \not\subseteq G \text{ for all } G \in \mathcal{G}. \tag{7.1}$$

When $\mathcal{F} = \mathcal{G}$, this problem is called the *saturation problem* in [EG95], where it is proved to be NP-complete. Given $\mathcal{F}, \mathcal{G} \subseteq 2^V$, we construct two families of cones $\mathcal{C}_\mathcal{F}$ and $\mathcal{C}_\mathcal{G}$ in $\mathbb{R}^V$, such that there is a point $x \in \mathbb{R}^V \setminus (\mathcal{C}_\mathcal{F} \cup \mathcal{C}_\mathcal{G})$ if and only if $\mathcal{F}$ and $\mathcal{G}$ are not saturated (i.e. there is a set $X \subseteq V$ satisfying (7.1)).

Informally speaking, we will interpret any point $x \in \mathbb{R}^V$ as subsets of $\{1, \cdots, V\}$ in one of the following two ways. In one interpretation x will represent the subset obtained by considering only those indices (coordinates) of x that are positive and in the other representation we will interpret x as representing the subset obtained by considering only negative coordinates. Let us call these two interpretations the positive and the negative interpretation, respectively.

For every hyperedge $F \in \mathcal{F}$ we will define polyhedral cones such that any point x lies in this cone if and only if its positive interpretation is a superset of F. Similarlly, for each $G \in \mathcal{G}$ we will define cones such that any point x lies in this cone if and only if the negative interpretation of x is a superset of $\overline{G}$. Thus if any point lies outside all of these cones then the positive interpretation of this point will give the desired X satisfying (7.1). The definition of such cones turns out be fairly straightforward, but they are usually not pointed and to make them pointed we will need to use Fact 1. Also, for brevity of notation we will denote the set of indices $\{1, \cdots, N\}$ simply by $[N]$.

For $X \subseteq V$, denote respectively by $\mathbb{R}^X_{\geq}$ and $\mathbb{R}^X_{\leq}$ the cones cone$\{\mathbf{e}_i : i \in X\} = \{x \in \mathbb{R}^X : \ x \geq 0\}$ and cone$\{-\mathbf{e}_i : i \in X\} = \{x \in \mathbb{R}^X : \ x \leq 0\}$, where $\mathbf{e}_i$ denotes the ith dimensional unit vector. Let $\overline{X} = V \setminus X$, and $\bigcup_{i=1}^{|X|+1} R_i(X) = \mathbb{R}^X$ be the partition of $\mathbb{R}^X$ given by Fact 1.

For each $F \in \mathcal{F}$, we define $|\overline{F}|+1$ cones $C^i_F = \mathbb{R}^F_{\geq} \times R_i(\overline{F})$, for $i \in [|\overline{F}|+1]$, and for each $G \in \mathcal{G}$, we define $|G|+1$ cones $C^i_G = \mathbb{R}^{\overline{G}}_{\leq} \times R_i(G)$, for $i \in [|G|+1]$. Finally, we let $\mathcal{C}_{\mathcal{F}} = \{C^i_F : F \in \mathcal{F}, \ i \in [|\overline{F}|+1]\}$, $\mathcal{C}_{\mathcal{G}} = \{C^i_G : G \in \mathcal{G}, \ i \in [|G|+1]\}$, and $\mathcal{C} = \mathcal{C}_{\mathcal{F}} \cup \mathcal{C}_{\mathcal{G}}$. Then it is not difficult to see that all the cones in $\mathcal{C}$ are pointed.

Suppose that $X \subseteq V$ satisfies (7.1). Define $x \in \mathbb{R}^V$ by

$$x_i = \begin{cases} 1, & \text{if } i \in X, \\ -1, & \text{if } i \in V \setminus X. \end{cases}$$

Then $x \notin \cup_{C \in \mathcal{C}} C$. Indeed, if $x \in C^i_F$, for some $F \in \mathcal{F}$ and $i \in [|\overline{F}|+1]$, then $x_i \geq 0$ and hence $x_i = 1$, for all $i \in F$, implying that $X \supseteq F$. Similarly, if $x \in C^i_G$, for some $G \in \mathcal{G}$ and $i \in [|G|+1]$, then $x_i \leq 0$ and hence $x_i = -1$, for all $i \in \overline{G}$, implying that $X \subseteq G$.

Conversely, suppose that $x \in \mathbb{R}^V \setminus \mathcal{C}$. Let $X = \{i \in V : \ x_i \geq 0\}$. Then we claim that X satisfies (7.1). Indeed, if $X \supseteq F$ for some $F \in \mathcal{F}$, then $x_i \geq 0$ for all $i \in F$, and hence there exists an $i \in [|\overline{F}|+1]$ such that $x \in C^i_F$ (since the cones $R_1(\overline{F}), \ldots, R_{|\overline{F}|+1}(\overline{F})$ cover $\mathbb{R}^{\overline{F}}$). Similarly, if $X \subseteq G$ for some $G \in \mathcal{G}$, then $x_i < 0$ for all $i \in \overline{G}$, and hence there exists an $i \in [|G|+1]$ such that $x \in C^i_G$. In both cases we get a contradiction. □

Corollary 7.3.6. *Problems* ConeCover$(\mathcal{H}, \mathbb{R}^d, \mathit{III})$, ConeCover$(\mathcal{H}, \mathbb{R}^d, \mathit{III})$ *and* ConeCover$(\mathcal{H}, \mathbb{R}^k, \mathit{III})$ *are NP-complete.*

Proof. NP-completeness of ConeCover$(\mathcal{V}, \mathbb{R}^d, \text{III})$ and ConeCover$(\mathcal{H}, \mathbb{R}^d, \text{III})$ follows from Theorem 7.3.5. NP-completeness of ConeCover$(\mathcal{H}, \mathbb{R}^k, \text{III})$ is an immediate consequence of the NP-hardness of ConeCover$(\mathcal{H}, \mathbb{R}^d, \text{III})$ and the fact that for an $\mathcal{H}$-cone, the intersection of this cone with any affine subspace can be computed easily. □

An interesting special case of problem Sat is when the two hypergraphs $\mathcal{F}$ and $\mathcal{G}$ are *transversal* to each other:

$$F \not\subseteq G \text{ for all } F \in \mathcal{F} \text{ and } G \in \mathcal{G}, \tag{7.2}$$

in which case, the problem is known as the *hypergraph transversal problem*, denoted HyperTrans. Even though the complexity of this problem is still open, it is unlikely to be NP-hard since there exist algorithms that solve the problem in quasi-polynomial time $m^{o(\log m)}$, where $m = |\mathcal{F}| + |\mathcal{G}| + |V|$. Improving this to a polynomial bound is a standing open question. We observe from our reduction in Theorem 7.3.5 that ConeCover includes HyperTrans as a special case.

Corollary 7.3.7. *Consider a family of cones $\mathcal{C}$ that can be partitioned into two families $\mathcal{C}_1$ and $\mathcal{C}_2$ such that*

$$\operatorname{int}(C_1) \cap \operatorname{int}(C_2) = \emptyset, \quad \textit{for all } C_1 \in \mathcal{C}_1 \textit{ and } C_2 \in \mathcal{C}_2. \tag{7.3}$$

Then ConeCover$(\mathcal{C}, \mathbb{R}^d)$ *is* HyperTrans*-hard.*

Proof. We note in the construction used on the proof of Theorem 7.3.5 that if the hypergraphs $\mathcal{F}$ and $\mathcal{G}$ satisfy (7.2), then the families of cones $\mathcal{C}_{\mathcal{F}}$ and $\mathcal{C}_{\mathcal{G}}$ satisfy (7.3). Indeed, if $x \in C_F^i \cap C_G^j$, for some $F \in \mathcal{F}$, $i \in [|\overline{F}|+1]$, $G \in \mathcal{G}$, and $j \in [|G|+1]$, then $x_k \geq 0$ for all $k \in F$ and $x_k \leq 0$ for all $k \in \overline{G}$. Thus for any $k \in F \setminus G$ (which must exist by (7.2)), we have $x_k = 0$, implying that x is not an interior point in either C_F^i or C_G^j. □

7.4 Covering sets by Polytopes

Freund and Orlin [FO85] proved that, for an $\mathcal{H}$-polytope P and a $\mathcal{V}$-polytope Q, checking if $Q \supseteq P$ is NP-hard. For all other representations of P and Q, checking $P \subseteq Q$ can be done by solving a linear program. Here we show that the union version of this problem is hard, no matter how the polytopes are represented.

Theorem 7.4.1. *Given a set of $\mathcal{H}$-polytopes $\mathcal{P} = \{P_1, \ldots, P_N\}$ and an $\mathcal{H}$ polytope P, problem* PolytopeCover$(\mathcal{P}, Q)$ *is NP-complete.*

Proof. Clearly the problem is in NP because if the given polytopes $\mathcal{P}$ do not cover P then there is a witness $x \in P$ such that $x \notin P_i$ for all $P_i \in \mathcal{P}$. This can easily be checked since we know the facets of each polytope. To prove that it is also NP-hard, we give a reduction from problem $\textsc{ConeCover}[\mathcal{H}, \mathbb{R}^d, \text{III}]$ which is NP-hard by Theorem 7.3.5.

Let S_d be a "shifted" simplex in $\mathbb{R}^d$ such that $\mathbf{0} \in \operatorname{int}(\{S_d\})$. Given cones $C_1, \ldots, C_N$, we define polytopes $P_1, \ldots P_N$, where $P_i = C_i \cap S_d$. Given the $\mathcal{H}$-representations of C_i, we can compute the $\mathcal{H}$-representations of P_i in polynomial time using linear programming (LP) for removing possible redundancies.

Now one can easily see that $\cup_{i=1}^N C_i = \mathbb{R}^d$ iff $\cup_{i=1}^N P_i = S_d$. $\square$

Theorem 7.4.2. *Given a set of $\mathcal{V}$-polytopes $\mathcal{P} = \{P_1, \ldots, P_N\}$ and a $\mathcal{V}$-polytope P, problem* $\textsc{PolytopeCover}(\mathcal{P}, Q)$ *is NP-complete.*

Proof. Similar to the proof of the previous theorem, it is easy to see that this problem is in NP. If the given polytopes $\mathcal{P}$ do not cover P completely, then there is a point in P that is not in any of the polytopes $P_i \in \mathcal{P}$. Checking whether a given point x lies in a $\mathcal{V}$-polytope can be easily done via linear programming. To show that this problem is also NP-hard, we give a reduction from problem $\textsc{ConeCover}[\mathcal{V}, \mathbb{R}^d, \text{III}]$ which is NP-hard by Theorem 7.3.5.

Recall that in the proof of Theorem 7.3.5, for each hyperedge F we construct a set of pointed cones $C_F^i = \mathbb{R}^F_{\geq} \times R_i(\overline{F})$, for $i \in [|\overline{F}| + 1]$. Instead of constructing multiple cones for each hyperedge let us just consider one cone $C_F = \mathbb{R}^F_{\geq} \times \mathbb{R}^{|\overline{F}|}$ per hyperedge. Similarly for the cones corresponding to the hypergraph $\mathcal{G}$. It is clear that $C_F = \cup_{i=1}^{|\overline{F}|+1} C_i$.

Note that each such cone is not pointed but instead has a pointed part $\mathbb{R}^F_{\geq}$ corresponding the the vertices in the hyperedge F and the affine space $\mathbb{R}^{|\overline{F}|}$ corresponding to the vertices not in F. Also, $\mathbb{R}^F_{\geq}$ is one orthant in $\mathbb{R}^{|F|}$.

For such cones checking whether the union covers $\mathbb{R}^d$ or not is NP-hard as well (See proof of Theorem 7.3.5). Now consider the d-dimensional cross polytope β_d. It contains the origin in its interior, and the vertices of $P_i = \beta_d \cap C_i$ for each cone constructed above can be easily computed. It is also easy to see that $\cup_{i=1}^k C_i = \mathbb{R}^d$ iff $\cup_{i=1}^k P_i = \beta_d$. $\square$

Theorem 7.4.3. *Given a set of rational convex polytopes $P_1, \ldots, P_k \subseteq \mathbb{Q}^d$, it is coNP-complete to check if their union is convex, for both $\mathcal{H}$ and $\mathcal{V}$-representations of the input polytopes.*

Proof. First we show that the problem is in coNP. Let $Q = \cup_{i=1}^k P_i$. If Q is not convex then there exist points $x, y \in Q$ and $\lambda \in (0, 1)$ such that the point

$\lambda x + (1-\lambda)y$ does not lie in any P_i. This can be trivially checked if we know the facets of each P_i, and amounts to linear programming if we know only the vertices of each P_i.

Now, consider the $\mathcal{H}$-representation first. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ and S_d be the polytopes used in the construction in theorem 7.4.1. We now reduce problem $\textsc{PolytopeCover}(\mathcal{P}, S_d)$ to checking if the union of a given set of polytopes is convex. Using an algorithm for the latter problem, we can check if $P = \cup_{i=1}^{k} P_i$ is convex. If the answer is "No", we conclude that $P \neq S_d$. Otherwise, since $P \subseteq S_d$, either $P = S_d$, or there is hyperplane separating a vertex of S_d from P. The latter condition can be checked in polynomial time by solving k linear programs for each vertex.

For the $\mathcal{V}$-representation the same argument as above works if we use the cross polytope β_d instead of S_d. $\square$

Chapter 8

Selfduality of Polytopes

The results in this chapter are a joint work with Khaled Elbassioni and have appeared in the proceedings of the 24th Symposium on Computational Geometry 2008 [TE08].

8.1 Introduction

In the previous chapters we considered many problems that are closely related to the vertex enumeration problem but that turn out to be NP-hard. Vertex Enumeration has a very interesting property in that if one tries to solve it by modifying the problem "a little bit", one runs into two kinds of problems. One kind are those that are polynomially equivalent to the original problem like polytope verification [ABS97]. Such problems do not really open up possibilities for any method that was not applicable to the original problem. The other kind of problems are ones for which a polynomial algorithm would yield a polynomial algorithm for VE but whose hardness has no consequence on VE. Examples include polytope containment [FO85], projection (Chapter 6), covering $\mathbb{R}^d$ with polyhedral cones (Chapter 7) and computing Minkowski sums (Chapter 5). Moreover, these problems usually turn out to be NP-hard and thus shed no light on the complexity of VE.

There is a similar ambiguity in the complexity status for graph isomorphism (GI) which, like the vertex enumeration problem, is not known to be either in P or to be NP-complete[1]. So therefore it is natural to try to relate the complexities of those two problems. The results in this chapter do not yet settle this issue, but we

[1]GI is known to be in $NP \cap coNP$ under certain assumptions ([MV99]).

make a step in this direction by deriving interesting connections between vertex enumeration and graph isomorphism.

For the purpose of relating vertex enumeration to graph isomorphism, we use the definition of a complexity class of all problems that are polynomially equivalent to GI. A problem Φ is said to be GI-easy if it can be solved in polynomial time given an oracle for GI and GI-hard if GI can be solved in polynomial time using an oracle for Φ. A problem that is both GI-easy and GI-hard is called GI-complete. Note that when talking about complexity of a problem with respect to an oracle, we assume that the oracle calls take constant (or polynomial) time. So for equivalence we are allowed to make a polynomial number of calls to the oracle.

In this chapter, we consider the problem of checking whether a polytope given by vertices or facets is combinatorially isomorphic to its polar dual. We will shortly clarify the meaning of the terms isomorphic. We call this problem Self-Duality problem (SD). SD is markedly different from the kind of problems mentioned earlier. We show that SD is GI-hard and furthermore that it is GI-complete if and only if VE is GI-easy. The "if and only if" in the result ensures that whichever way the complexity of SD is settled, it will have non-trivial consequences for the complexity of VE. To the best of our knowledge this is the first problem that opens up the possibility of relating the complexity of VE to that of GI.

For a polytope P, recall that $\mathcal{V}(P)$ and $\mathcal{F}(P)$ denote the sets of vertices and facets respectively. The *facet-vertex incidence matrix* $\mathcal{I}(P) \in \{0,1\}^{m\times n}$, of a polytope P with $m = |\mathcal{F}(P)|$ and $n = |\mathcal{V}(P)|$, is a 0/1-matrix whose rows represent the facets and whose columns represent the vertices and $\mathcal{I}[i,j] = 1$ if and only if the i-th facet is incident to the j-th vertex. It is known that the face lattice of a polytope is completely determined by its facet-vertex incidence matrix ([KP02]). A real matrix A is said to be *transposable* if it can be transformed into its transpose A^T via row and column permutations, and is said to be *symmetrizable* if it can be transformed into a symmetric matrix via row and column permutations. For a nice survey of transposability and symmetrizability of matrices the reader is referred to [BGZ06].

Two polytopes P and Q are said to be *combinatorially isomorphic* to each other, denoted by $P \cong Q$, if their face-lattices are isomorphic. For example, any two convex polygons with equal number of sides are combinatorially isomorphic. Equivalently, two polytopes are isomorphic if and only if the incidence matrix of one can be transformed into that of the other via row and column permutations. A polytope is said to be *self-dual* if it is combinatorially isomorphic to its polar dual, i.e., if $P \cong P^*$. In terms of incidence matrices this means that for self-dual polytopes the incidence matrix is transposable. Also, the row and column permutation that changes the incidence matrix to its transpose is called the self-duality map. Note that the incidence matrix of a self-dual polytope need not be symmetrizable (See [Jen89]).

8.2 Related Work

Our work touches on various topics including vertex enumeration, isomorphism and self-duality of polytopes, as well as transposability and symmetrizability of 0/1-matrices. In this section, we will briefly mention some of the existing literature pertaining to these topics.

Self-dual polytopes form an interesting subclass of polytopes and their classification is a fundamental problem in the theory of polytopes. Self-dual polytopes have been studied extensively at least in 3-dimensions and the 3-dimensional spherical and projective self-dual polytopes have been fully characterized (See [AN93]). In higher dimensions not much appears to be known. As we will see, the free-join of a polytope P and its polar dual[2] always generates self-dual polytopes. Also, the free-join of any two self-dual polytopes yields another self-dual polytope. These constructions do not yield all possible self-dual polytopes but the ones that do arise have interesting properties, namely that they also admit an involutory self-duality map. In section 8.5, we describe[3] a class of polytopes which we call *roofed-prisms* that are self-dual but are not obtainable as free-join of simpler polytopes.

Kaibel and Schwartz ([KS03]) studied various isomorphism questions about polytopes and proved that it is GI-complete to determine if two polytopes given by their facet-vertex incidences are combinatorially isomorphic to each other. The problem remains GI-complete even if the coordinates of vertices and facet-normals are provided or if the polytopes are restricted to be simple or simplicial polytopes. The authors in [KS03], however, leave open the question of checking self-duality

[2]Instead of the polar dual one can use any polytope combinatorially isomorphic to the polar P^*.

[3]We do not use these polytopes in our proofs but the construction is simple enough to warrant mentioning these polytopes in this context.

of a polytope given by its facet-vertex incidence matrix. This problem (called SDI from now on) is a variant of SD[4] and in the process of relating SD to VE, we settle the question of Kaibel and Schwartz by proving that SDI is GI-complete as well.

As we noted before, self-duality of a polytope implies that its incidence matrix is transposable. Also, if the self-duality map is involutory, *i.e.* the map applied twice yields the identity map, then the incidence matrix is symmetrizable. Note that every symmetrizable matrix is transposable, but there are transposable matrices that are not symmetrizable [BGZ06]. Grünbaum asked in [GS88] if there are self-dual polytopes that do not have any involutory self-duality maps. Jendrol [Jen89] answered this question in the affirmative.

We should remark that we do not claim the novelty of the constructions provided in this chapter. Free join of two polytopes is a well known operation ([Zie95]). The same construction is also attributed to David Eppstein [Eri00] in finding examples of polytopes with n vertices, n facets and $n^{\lfloor \frac{d+1}{3} \rfloor}$ faces improving an earlier bound of $n^{\sqrt{d}}$ by Seidel *et al.* [Eri00, ABS97]. The author is not aware of any other work mentioning the roofed-prisms, that are mentioned in this chapter, as an example of indecomposable self-dual polytopes.

8.3 Main Results

In this chapter we consider the following three problems:

VE: Given a polytope P by facets, enumerate all the vertices of P.
SD: Given a polytope P by facets *or* vertices, determine if P is self-dual.
SDI: Given a polytope P by its facet-vertex Incidence Matrix, determine if P is self-dual.

Our main results are the following:

- SD is GI-hard.
- SD is GI-complete if and only if VE is GI-easy.
- SDI is GI-complete.

For proving the GI-hardness of SD and its relations to the complexity of VE, we start by exploring the complexity of SDI. We establish that SDI is GI-complete first and the other results are easy consequences of this fact. Our results on the

[4]Recall that for SD the polytope is given only by its facets or only vertices.

complexity of SDI strengthens the result of [KS03] that it is GI-complete to determine if two polytopes given by their facet-vertex incidences are combinatorially isomorphic to each other. We arrive at this result by showing that, essentially the *free join* of two polytopes is self-dual if and only if the two polytopes are isomorphic.

8.4 Constructing Self Dual Polytopes

8.4.1 Free Join

Two affine spaces are called *skew* if they neither intersect nor contain any parallel lines. The *free join* of two polytopes is obtained by embedding the polytopes in skew subspaces and taking the convex hull. For example, the free join of two line segments is a 3-dimensional tetrahedron. Since in the context of this chapter we are interested only in the combinatorial structure of polytopes arising as free-joins of smaller polytopes independent of the actual embedding, we will choose some specific skew hyperplanes for the purpose of embedding the component polytopes. Let P_1 and P_2 be two polytopes in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively, such that:

$$\begin{aligned} P_1 &= \{x \in \mathbb{R}^m \mid A_1 x \leq 1\} = \mathrm{conv}(V_1), \\ P_2 &= \{x \in \mathbb{R}^n \mid A_2 x \leq 1\} = \mathrm{conv}(V_2), \end{aligned}$$

where $A_1 \in \mathbb{R}^{l \times m}$, $V_1 \subseteq \mathbb{R}^m$, $A_2 \in \mathbb{R}^{r \times n}$, and $V_2 \subseteq \mathbb{R}^n$, then the vertices of the free join $P * Q \subseteq \mathbb{R}^{m+n+1}$ are

$$\mathcal{V}(P_1 * P_2) = \left\{ \begin{pmatrix} v \\ \mathbf{0} \\ -1 \end{pmatrix} : v \in \mathcal{V}(P_1) \right\} \bigcup \left\{ \begin{pmatrix} \mathbf{0} \\ v \\ 1 \end{pmatrix} : v \in \mathcal{V}(P_2) \right\}$$

and

$$P_1 * P_2 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} : 2A_1 x + z \cdot \mathbf{1}_l \leq \mathbf{1}_l,\ 2A_2 y - z \cdot \mathbf{1}_r \leq \mathbf{1}_r \right\},$$

where $\mathbf{1}_k$ is a vector in $\mathbb{R}^k$ all whose entries are 1. The following are some easy observations about the free join operation.

Fact 2. *Suppose P_1 is an i-dimensional polytope and P_2 is a j-dimensional polytope. If $P = P_1 * P_2$, then*

(i) An i-dimensional face of P is the free join of an r-dimensional face of P_1 with an s-dimensional face of P_2 such that $r + s + 1 = i$, and consequently,

(ii) every face of P_1 or P_2 is a projection of a face of P, with the same dimension,

(iii) P is an $(i + j + 1)$-dimensional polytope,

(iv) $|\mathcal{V}(P)| = |\mathcal{V}(P_1)| + |\mathcal{V}(P_2)|$, and

(v) $|\mathcal{H}(P)| = |\mathcal{H}(P_1)| + |\mathcal{H}(P_2)|$. Furthermore, every facet of P is either a free join of P_1 and some facet of P_2, or that of P_2 and some facet of P_1.

8.4.2 Incidence Matrix of Free-Join

Recall that the facet-vertex incidence matrix $\mathcal{I}(P)$ of a polytope P has facets as rows and vertices as columns and the (i, j)-th entry is 1 iff the i-th facet contains the j-th vertex. In particular, if P is full-dimensional, then no row or column of $\mathcal{I}(P)$ can consist of all ones. It follows that the incidence matrix of a polytope $P = P_1 * P_2$, that is a free join of two polytopes P_1 and P_2, is of the form $\left[\frac{A|B}{C|D}\right]$ where A and D are submatrices all whose entries are 1's, and B and C are the incidence matrices of P_1 and P_2. The following lemma establishes that for the incidence matrix of a polytope to be decomposable into the aforementioned form, it is also necessary that the polytope be a free-join of other (simpler) polytopes.

Lemma 8.4.1. *Let P be a full-dimensional polytope in $\mathbb{R}^d$. Under suitable labeling of vertices and facets, the incidence matrix $\mathcal{I}(P)$ of P is of the form $\left[\frac{A|B}{C|D}\right]$, where A and D are submatrices all of whose entries are 1's, if and only if P is a free join of two polytopes P_1 and P_2 with respective incidence matrices B and C.*

Proof. If P is a free join of two polytopes, then the incidence matrix of P can be written in the desired form, as explained above.

Now, suppose that the incidence matrix of P is of the required form. Suppose, the dimensions of the matrices A, B, C, D are $m_1 \times n_1$, $m_1 \times n_2$, $m_2 \times n_1$ and $m_2 \times n_2$, respectively. Let the set of vertices corresponding to the first n_1 columns of $\mathcal{I}(P)$ be V_1 and the ones corresponding to the last n_2 columns be V_2. Similarly, let F_1 and F_2 be the sets of facets corresponding to the first m_1 and last m_2 rows respectively.

Since any affine transformation preserves incidences, we can assume that P contains the origin in the interior. Suppose that the halfspaces corresponding to the facets F_1 of P be $A_1 x \leq 1$, and the halfspaces corresponding to the facets F_2 of P be $A_2 x \leq 1$.

Note that no row or column of B or C has all 1's, since otherwise $\mathcal{I}(P)$ has such a row or a column. This implies that the affine hull of V_1 can be obtained as the intersection of the hyperplanes defining the facets in F_1 and that of V_2 can be obtained as the intersection of the hyperplanes defining F_2. Specifically, the affine hull of V_1 is $\{x | A_1 x = 1\}$ and that of V_2 is $\{x | A_2 x = 1\}$.

Since P is full dimensional polytope, there is no common intersection for all the hyperplanes defining $F_1 \cup F_2$. (Indeed, if x is a point in such intersection and $x' \in \text{int}(P)$, then the ray starting at x and moving through x' must hit P at some facet $F \in \mathcal{F}(P)$ whose defining hyperplane contains x. But this would imply that the whole ray belongs to this hyperplane, and hence that $x' \in F$, in contradiction to the fact that x' is an interior point in P.) Hence, the affine hulls of V_1 and V_2 don't intersect.

Now suppose that the affine hulls of V_1 and V_2 are not skew, i.e. they contain parallel lines. Let the copy of this parallel line in the affine hull of V_1 have the parametric equation $l_1 = \{x | x = \alpha_1 + t \cdot u, t \in \mathbb{R}\}$, where $x, \alpha_1, u \in \mathbb{R}^d$. Similarly let the copy in the affine hull of V_2 be $l_2 = \{x = \alpha_2 + t \cdot u, t \in \mathbb{R}\}$. Note that since the two copies are parallel to each other their "direction" is defined by the same vector u.

Since l_1 lies in the affine space $A_1 x = 1$, we have $A_1 \cdot (\alpha_1 + t \cdot u) = 1$ for all values of t. This means $A_1 \cdot u = 0$. Similarly it follows from l_2 that $A_2 \cdot u = 0$. But A_1 and A_2 cover all rows of A and so $A \cdot u = 0$. Clearly for l_1 and l_2 to be lines u must not be the zero vector. But if $A \cdot x = 0$ has a non-trivial solution u then $A \cdot (\lambda u) = 0 \leq 1,\ \forall \lambda \in \mathbb{R}$. This contradicts our assumption that P is a bounded polytope and hence does not contain any lines.

Hence, the affine hulls of V_1 and V_2 are skew and P is the free-join of the two polytopes defined by these two sets of vertices. □

8.5 Complexity of SDI and SD

Our starting point is the following result of Kaibel and Schwartz [KS03]: Given two polytopes P_1 and P_2 by their vertices and facets, it is GI-complete to determine whether they are isomorphic to each other. In fact, this is true even if each polytope P_i satisfies the following conditions, for $i \in \{1, 2\}$:

(C1) P_i is *simple*, i.e., every vertex of P_i lies on exactly d facets, where $d = \dim(P_i)$,

(C2) $|\mathcal{V}(P_i)| + 2 \neq 2|\mathcal{F}(P_i)|$, and

(C3) $|\mathcal{F}(P_i)| > 2\dim(P_i)$.

(More precisely, the reduction in [KS03] constructs for a graph $G = (V, E)$ a simple polytope $P(G)$ of dimension $d = |V| - 1$, with $|\mathcal{V}(P(G))| = |V|(|V| - 1) + 2|E|(|V| - 2)$ and $|\mathcal{F}(P(G))| = 2|V| + 2|E|$. In particular, $|\mathcal{V}(P(G))| + 2 > 2|\mathcal{F}(P(G))|$ for $|V| \geq 5$, i.e., (C1), (C2), and (C3) are satisfied.)

Before we proceed with the details of our reduction, we need the following definition. Given a full-dimensional polytope $P \in \mathbb{R}^d$, a $(d+1)$-dimensional *bipyramid* $\text{bipyr}(P)$, constructed from P, is obtained by taking two points $u, v \in \mathbb{R}^{d+1}$, strictly in two different sides of $\text{aff}(P)$, such that the line segment connecting u and v intersects the relative interior of P, and defining $\text{bipyr}(P) = \text{conv}(P \cup \{u, v\})$. P is called the base of the bipyramid and u, v are called the *apices*.

Our reduction of GI to SDI works as follows: Given two graphs G_1 and G_2, we first construct polytopes P_1 and P_2 as in Kaibel and Schwartz ([KS03]). Next we consider the polytope P obtained by taking the free join of $\text{bipyr}(P_1)$ with the polar dual of $\text{bipyr}(P_2)$. We show that under assumptions (C1), (C2) and (C3) P is self-dual if and only if G_1 and G_2 are isomorphic.

Lemma 8.5.1. *Let P be a full-dimensional polytope in $\mathbb{R}^d$, $d \geq 3$, with m facets and n vertices such that $n + 2 \neq 2m$. Then the bipyramid $Q = \text{bipyr}(P)$ is neither a self-dual polytope nor can it be obtained as the free join of two other polytopes.*

Proof. The bipyramid Q has $2m$ facets and $n + 2$ vertices. Since Q has unequal number of vertices and facets it is clearly not self-dual.

To prove that Q is not decomposable as free-join of smaller polytopes, consider the two apices of Q. Suppose Q is decomposable as $P_1 * P_2$. Since every pair of vertices from P_1 and P_2 generates an edge in the free-join, both apices of Q must be part of one of the component polytopes, say P_1 *wlog*. But then both apices must lie in a proper face of Q contradicting the fact that they are the apices of a bipyramid. □

Lemma 8.5.2. *Let P_1 and P_2 be two polytopes satisfying (C1) and (C3). Then $P_1 \cong P_2$ if and only if* $\operatorname{bipyr}(P_1) \cong \operatorname{bipyr}(P_2)$*.*

Proof. If $P_1 \cong P_2$, then clearly $\operatorname{bipyr}(P_1) \cong \operatorname{bipyr}(P_2)$. Suppose now that $\operatorname{bipyr}(P_1) \cong \operatorname{bipyr}(P_2)$. Then there is an order-preserving bijection ϕ between the face lattices of $\operatorname{bipyr}(P_1)$ and $\operatorname{bipyr}(P_2)$. Let u_i, v_i be the apices of $\operatorname{bipyr}(P_i)$, $d_i = \dim(P_i)$, $n_i = |\mathcal{V}(P_i)|$, and $m_i = |\mathcal{F}(P_i)|$, for $i \in \{1, 2\}$. Since $\operatorname{bipyr}(P_1) \cong \operatorname{bipyr}(P_2)$, we have $d_1 = d_2 = d$, $n_1 = n_2 = n$, and $m_1 = m_2 = m$. For a point $u \in \mathbb{R}^d$ and a polytope P, we denote by $f(P, u)$ the number of facets of P containing u. Then $f(P_i, u) = d$ for all $u \in \mathcal{V}(P_i)$ follows from the simplicity of P_i, for $i \in \{1, 2\}$. Thus, $f(\operatorname{bipyr}(P_i), u_i) = f(\operatorname{bipyr}(P_i), v_i) = m$, while $f(\operatorname{bipyr}(P_i), u) = 2d$ for all $u \in \mathcal{V}(\operatorname{bipyr}(P_i)) \setminus \{u_i, v_i\}$, for $i \in \{1, 2\}$. Since $m > 2d$ by (C3), it follows that $\phi(\{u_1, v_1\}) = \{u_2, v_2\}$. Then the restriction of ϕ on the faces of P_1 gives an isomorphism between the face lattices of P_1 and P_2. □

Recall that a matrix is called transposable if its rows and columns can be permuted to obtain its transpose, and is called symmetrizable if it can be converted into a symmetric matrix by row and column permutations.

Fact 3. *A polytope is self-dual if and only if its incidence matrix is transposable.*

With these notions, we are ready to establish the following result.

Theorem 8.5.3. *Let P_1, $P_2 \in \mathbb{R}^d$ be two polytopes, neither of which is self-dual or decomposable into a free join of other polytopes. Then, $P_1 * P_2$ is self-dual if and only if $P_1 \cong P_2^*$.*

Proof. For $i \in \{1, 2\}$, let V_i, F_i and A_i be respectively the set of vertices, set of facets, and incidence matrix of P_i, and write $n_i = |\mathcal{V}(P_i)|$ and $m_i = |\mathcal{F}(P_i)|$. Then the incidence matrix of $P_1 * P_2$ is of the form shown in Figure 8.1-(a).

If P_1 is isomorphic to P_2^*, then $m_1 = n_2 = m$, $n_1 = m_2 = n$, and there exist row and column permutations σ_1, ρ_1 for A_1 that transform it to A_2^T and also, there exist row and column permutations σ_2, ρ_2 for A_2 that transform it to A_1^T. Now

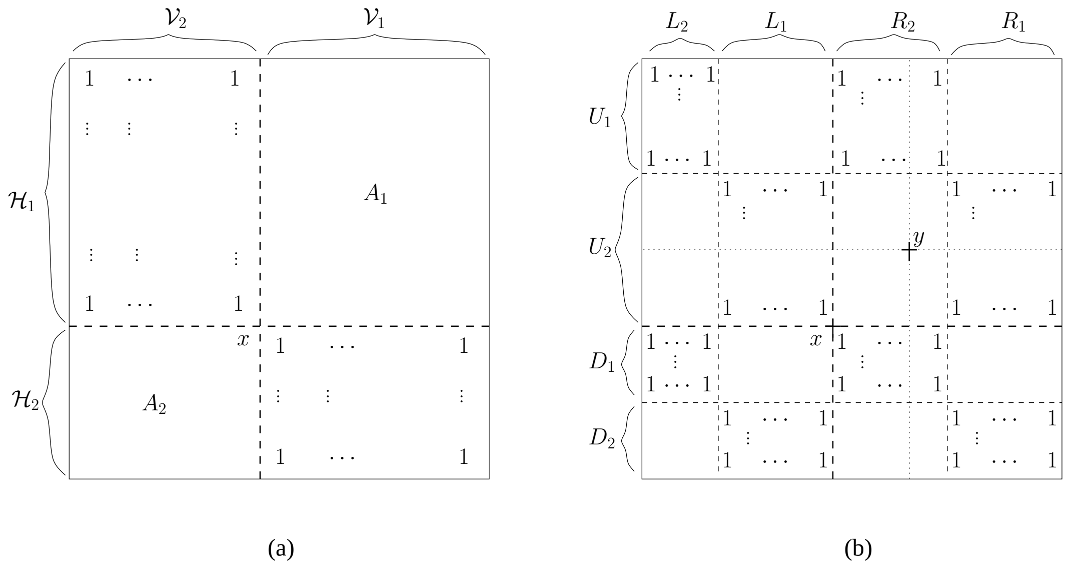


Figure 8.1: The incidence matrix $C = \mathcal{I}(P_1 * P_2)$ before and after applying the permutations σ and ρ. In (b), the two dotted lines, crossing at y, indicate the partition of C^T resulting from the original partition of C. In particular, the upper-right corner above y contains the matrix A_2^T, while the lower-left corner contains A_1^T.

consider the following permutations σ, ρ of the rows and columns for the incidence matrix of $P_1 * P_2$ (assume the vertices of $P_1 * P_2$ are numbered $1, 2, 3, \ldots$, and similarly the facets):

$$\sigma(i) = \begin{cases} \sigma_1(i) & \text{if } i \leq m, \\ m + \sigma_2(i - m) & \text{if } i > m. \end{cases}$$

$$\rho(i) = \begin{cases} \rho_2(i) & \text{if } i \leq m, \\ m + \rho_1(i - m) & \text{if } i > m. \end{cases}$$

It is easy to see that this permutation of rows and columns applied to the incidence matrix of $P_1 * P_2$ produces its transpose and hence $P_1 * P_2$ is self-dual.

Now, to prove the other direction, assume that $P_1 * P_2$ is self-dual. Then $m_1 + m_2 = n_1 + n_2$, and there exist row and column permutations, σ, ρ, of the incidence matrix $C = \mathcal{I}(P_1 * P_2)$ that transform it to its transpose. Assume w.l.o.g. that $m_1 \geq n_2$ and hence $n_1 \geq m_2$. Define the following subsets of row and column indices according to σ and ρ:

$$L_1 = \{i \mid i \leq n_2,\ \rho(i) > n_2\}\,,\ L_2 = \{i \mid i \leq n_2,\ \rho(i) \leq n_2\}\,,$$

$$R_1 = \{i \mid i > n_2,\ \rho(i) > n_2\}\,,\ R_2 = \{i \mid i > n_2,\ \rho(i) \leq n_2\}\,,$$

$$U_1 = \{i \mid i \leq m_1,\ \sigma(i) \leq m_1\}\,,\ U_2 = \{i \mid i \leq m_1,\ \sigma(i) > m_1\}\,,$$

$$D_1 = \{i \mid i > m_1,\ \sigma(i) \leq m_1\}\,,\ D_2 = \{i \mid i > m_1,\ \sigma(i) > m_1\}\,.$$

In other words, if we call the initial columns corresponding to $\mathcal{V}_2$ *left* columns and those corresponding to $\mathcal{V}_1$ *right* columns then L_1 corresponds to the set of vertices of $\mathcal{V}_1$ that moves to the left after applying ρ and R_1 corresponds to the set of vertices of $\mathcal{V}_1$ that remains in the right. Similarly, if we call the rows *up* or *down* depending on whether they correspond to facets $\mathcal{F}_1$ or $\mathcal{F}_2$ respectively, then U_1 corresponds to the subset of $\mathcal{F}_1$ that remains up after the column permutation σ and D_1 corresponds to the subset that moves down (see Figure 8.1-(b)).

Since ρ, σ transform C to C^T, it follows from the definitions of the above sets that $C^T[i,j] = 1$ for $(i,j) \in U_1 \times L_2, U_1 \times R_2, U_2 \times L_1, U_2 \times R_1, D_1 \times L_2, D_1 \times R_2, D_2 \times L_1, D_2 \times R_1$ (see Figure 8.1-(b)).

We claim that $|U_1| + |U_2| = |L_1| + |L_2|$, or in other words, the two points x and y in Figure 8.1-(b) coincide. If this was not the case, then the point y would lie in one of the four possible corners $U_1 \times R_2$, $U_1 \times R_1$, $U_2 \times R_2$, or $U_2 \times R_1$. Consider w.l.o.g. the situation in Figure 8.1-(b), where $y \in U_2 \times R_2$. Since the submatrix of C^T above and to the right of y is A_2^T, it follows from Lemma 8.4.1 that the polytope P_2^* is decomposable, in contradiction to our assumptions. Similarly, in all the other three cases for y, one can verify that there exist row and column permutations of A_1^T, such that the resulting matrix, and hence P_1^*, have a decomposition in the sense of Lemma 8.4.1.

Thus both $|U_1| + |U_2|$ and $|L_1| + |L_2|$ are equal to, say m, and hence transposing C gives $C^T[i,j] = 1$ for all $(i,j) \in (U_1 \cup U_2) \times (L_1 \cup L_2)$. However, C^T is also obtained by transforming C using ρ, σ, and thus we get $C[i,j] = 1$ for $(i,j) \in L_1 \times U_1, D_1 \times R_1$. Since P_1 is indecomposable, it follows from Lemma 8.4.1 that either $L_1 = D_1 = \emptyset$ or $R_1 = U_1 = \emptyset$ (any other choice would give an all 1's row or column in C). The latter case would imply that A_1^T is mapped by row and column permutations into A_1^T, and hence is not possible, since P_1 is assumed not to be self-dual. Hence the permutations σ, ρ leave the vertices of $\mathcal{V}_1$ and the facets of $\mathcal{F}_1$ in their own blocks. A similar argument can be made about the rows and columns corresponding to $\mathcal{V}_2$ and $\mathcal{F}_2$. Hence, the permutations σ, ρ satisfy the following:

$$\begin{aligned} \rho(i) \leq m &\quad \text{iff } i \leq m \\ \sigma(i) \leq m &\quad \text{iff } i \leq m \end{aligned}$$

Now we can define a permutation of rows $\sigma^{'}$ and columns $\rho^{'}$ of the incidence matrix A of P as follows:

$$\begin{aligned} \sigma^{'}(i) = \sigma(i) &\qquad \text{for } i = 1 \ldots, m \\ \rho^{'}(i) = \rho(m+i) - m &\qquad \text{for } i = m+1 \ldots, m+n \end{aligned}$$

This transforms A_1 into A_2^T and hence shows that P_1 is isomorphic to the dual of P_2. □

We remark that actually one need not assume that the polytopes in the previous theorem are not self-dual and the following stronger version of Theorem 8.5.3 is true:

Theorem 8.5.4. *Let P_1, $P_2 \in \mathbb{R}^d$ be two polytopes that are not decomposable into free join of other polytopes. Then, the free join $P_1 * P_2$ is self-dual if and only if either $P_1 \cong P_2^*$ or both P_1 and P_2 are self-dual.*

This theorem can be proved with only a slight modification of the proof of Theorem 8.5.3 but for the purposes of our proof of GI-completeness of SDI, we need the polytopes to not be self-dual and so we will keep working with the weaker version of the theorem. Also, it follows from the proof of Theorem 8.5.3 that although the notion of transposability and symmetrizability of general 0/1-matrices are different, for the incidence matrices of the self-dual polytopes that arise from Theorem 8.5.3, both notions are equivalent.

Corollary 8.5.5. *If P and Q are two polytopes in $\mathbb{R}^d$ such that both P and Q are neither decomposable nor self-dual, then P is isomorphic to Q if and only if the incidence matrix $P * Q^*$ is symmetrizable.*

Corollary 8.5.6. *For a polytope P, $P * P^*$ is self-dual and the incidence matrix of the free join is symmetrizable.*

Now, we can state the final theorem of this section.

Theorem 8.5.7. *Let P be a polytope in $\mathbb{R}^d$ given by its facet-vertex incidence matrix or both vertices and facets. It is GI-complete to determine whether P is self-dual.*

Proof. Clearly, if an oracle for GI is given then it can be used to check self-duality of a polytope given by its incidence matrix simply by checking if $P \cong P^*$. Since the incidence matrix can also be computed from the vertices and facets in polynomial time, checking self-duality in this case is GI-easy.

To show that the self-duality checking is also GI-hard, we use the following GI-complete problem [KS03]: Given two polytopes P_1 and P_2 by their facet and vertex descriptions, or by their facet-vertex incidence matrix, determine if $P_1 \cong P_2$.

As mentioned at the beginning of section 8.5, we may assume that P_1 and P_2 satisfy conditions (C1), (C2) and (C3). From the vertices, facets or facet-vertex incidence matrices of P_1 and P_2, we can construct, in polynomial-time, the vertices, facets or incidence matrix (resp.) of $P = \text{bipyr}(P_1) * \text{bipyr}(P_2)^*$. By Lemma 8.5.1, both $\text{bipyr}(P_1)$ and $\text{bipyr}(P_2)^*$ are neither self-dual nor decomposable as free-joins of other polytopes. By Theorem 8.5.3, P is self-dual if and only if $\text{bipyr}(P_1) \cong \text{bipyr}(P_2)$, and by Lemma 8.5.2, the latter condition is equivalent to $P_1 \cong P_2$. $\square$

Recall that for problem SD we want to verify whether a polytope, given by only vertices or only facets, is self-dual. An easy corollary of Theorem 8.5.7 is that SD is GI-hard.

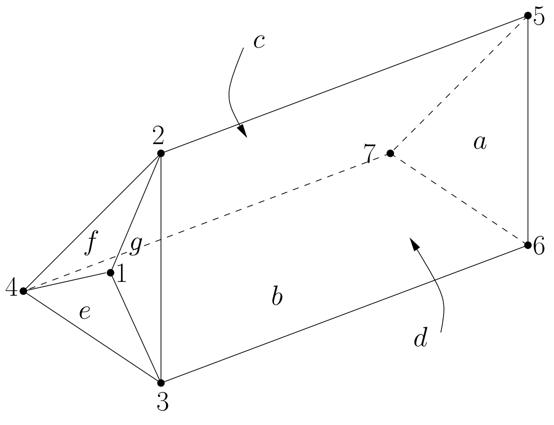
Corollary 8.5.8. *Let P be a polytope in $\mathbb{R}^d$ given by its vertices (or facets). It is GI-hard to determine whether P is self-dual.*

In the next section, we will discuss some interesting consequences of the complexity of SD on the problem of enumerating vertices of a polytope given by its facets.

We conclude this section by remarking that not all self-dual polytopes arise from free-join of other "smaller" self-dual polytopes. For instance, Figure 8.2 shows an example of a 3-dimensional polytope which is self-dual but indecomposable in the sense of free-join. This example can be generalized to yield an infinite family of indecomposable self-dual polytopes, which we call *roofed-prisms*, as follows. Let P be a d-dimensional polytope and $u, v \in \mathbb{R}^{d+1}$ be two points strictly in two

$$\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
a & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
b & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
c & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
d & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
e & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
f & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
g & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
\end{array}$$

(a)



(b)

Figure 8.2: A 3-dimensional self-dual indecomposable polytope and its incidence matrix.

different sides of $\text{aff}(P)$, such that the line segment connecting u and v intersects the interior of P. Let P' be a parallel copy of P containing u and define $Q(P) = \text{conv}(P \cup P' \cup \{v\})$. Informally, $Q(P)$ is obtained by putting the pyramid of P as a roof on the (vertical) prism of P. Then for any self-dual polytope P, the roofed-prism $Q(P)$ is self-dual and indecomposable (a fact that can be proved using Lemma 8.4.1.)

8.6 Vertex Enumeration

As noted in the introduction, a problem that is polynomially equivalent to vertex enumeration is the problem of determining whether an $\mathcal{H}$-polytope P is the same as a $\mathcal{V}$-polytope Q [ABS97], also known as polytope verification. Clearly, we may assume that $\mathcal{V}(Q) \subseteq \mathcal{V}(P)$, and furthermore that $\{\text{aff}(F) \mid F \in \mathcal{F}(P)\} \subseteq \{\text{aff}(F) \mid F \in \mathcal{F}(Q)\}$, for otherwise, P and Q can not be the same. The following theorem relates this problem (and hence VE) to the problem of checking self-duality of a given polytope.

Theorem 8.6.1. *Let $P \subset \mathbb{R}^d$ be an $\mathcal{H}$-polytope and $Q \subset \mathbb{R}^d$ be a $\mathcal{V}$-polytope such that $\mathcal{V}(Q) \subseteq \mathcal{V}(P)$ and $\{\text{aff}(F) \mid F \in \mathcal{F}(P)\} \subseteq \{\text{aff}(F) \mid F \in \mathcal{F}(Q)\}$. Then, $P = Q$ if and only if $P * Q^*$ is self-dual.*

Proof. It is easy to see that if $P = Q$ then $P * Q^*$ is self-dual. On the other hand, if $P \neq Q$ then $|\mathcal{V}(P)| > |\mathcal{V}(Q)|$ and also $|\mathcal{F}(Q)| > |\mathcal{F}(P)|$. Hence, $|\mathcal{F}(P * Q^*)| = |\mathcal{F}(P)| + |\mathcal{V}(Q)| < |\mathcal{F}(Q)| + |\mathcal{V}(P)| = |\mathcal{V}(P * Q^*)|$. Thus $P * Q^*$ has strictly fewer facets than vertices and hence it can not be self-dual. □

As we have seen in the previous section SD is GI-hard. Now there are two possibilities: either SD is really harder than GI in that there is a strict (non-polynomial) gap between the complexities of SD and GI, or SD is in fact GI-easy and hence GI-complete as well. In both cases, we get a similar statement about the complexity of VE.

Theorem 8.6.2. *VE is GI-easy if and only if SD is GI-complete.*

Proof. Clearly, if SD is GI-easy then VE is GI-easy, since an oracle for GI would solve SD, which in turn would solve polytope verification, by Theorem 8.6.1. On the other hand, suppose that VE is GI-easy, and suppose we are given an instance of SD, i.e., a polytope P described by, say, its facets, and we want to check whether P is self-dual. Using the oracle for GI, we can enumerate the vertices of P. If P has too many vertices, we know that it is not self-dual and if the number of vertices of P is equal to the number of facets of P, then after enumerating vertices of P we have both vertex and facet descriptions of P, and now the self-duality can be checked using an oracle for GI. Since we know SD to be GI-hard by Corollary 8.5.8, SD is also GI-complete. □

Since GI is not believed to be NP-hard, by Theorem 8.6.2, if SD is GI-easy, then VE is probably also not NP-hard.

Chapter 9

Conclusion

9.1 Thesis Summary

Any polytope $P \subset \mathbb{R}^d$ can be represented as either the convex hull of a finite number of points in $\mathbb{R}^d$ or as a bounded intersection of a finite number of halfspaces. Either of these representations defines the other uniquely if redundancies are not allowed, but no output-sensitive algorithms are known that convert one of these representations to the other. Also, depending on whether the vertex representation or the facet representation of a polytope is known, the computational complexity of an operation can vary wildly. This thesis considers various computational problems about polytopes and their complexity.

The problems that we consider in this thesis, and the results that we obtain, have the following common themes:

- Many of the problems considered in this thesis are very fundamental operations on polytopes. Operations like the intersection, Minkowski addition and projection are very basic operations on polytopes that need to be frequently computed in practice. The complexity results in this thesis show that often the best way to perform these operations is by converting the representation of the input polytope first, highlighting the importance of finding an output-sensitive vertex enumeration algorithm.

- Every problem considered in this thesis is closely related to the vertex enumeration problem. For most problems, an efficient algorithm would imply and output-sensitive algorithm for vertex enumeration. Unfortunately, most of these problems turn out to be NP-hard or $\#P$-hard. The problem of checking whether a polytope is isomorphic to to its polar dual (chapter 8), on the other hand, relates the vertex enumeration problem with the well know graph isomorphism problem. Admittedly though the connection established in this thesis is somewhat weak.

- Every problem highlights the impact of input representation for computational problems about polytopes. For example, computing the vertex centroid (chapter 4) for a $\mathcal{V}$-polytope is trivial but is $\#P$-hard for $\mathcal{H}$-polytopes. Similarly, computing the facets of the intersection of two $\mathcal{H}$-polytopes is easy but the problem is NP-hard for almost all other representations.

The main contributions of this thesis are roughly divided into different categories as mentioned in the following subsections.

9.1.1 Strengthening previous results

In chapter 3 we show that enumerating the vertices of an unbounded $\mathcal{H}$-polyhedron P is NP-hard even when P has only 0/1 vertices. This strengthens a previous result of Khachiyan et. al. [KBB$^+$06] for general polyhedra. For general $\mathcal{H}$-polytopes the complexity of enumerating all the vertices is unknown but for a polytope all whose vertices are 0/1 there exists an output-sensitive algorithm [BL98]. Our result, thus, provides a better contrast between the complexities of vertex enumeration problem for polytopes and unbounded polyhedra.

Our small but very crucial modifications of the proof in [KBB$^+$06] allows us to obtain various other hardness results. For example, among other things, our method shows that identifying whether an $\mathcal{H}$-polyhedron is a 0/1-polyhedron or not is possible in polynomial time unless $P = NP$. Many of these additional results in chapter 3 are already known but our method allows us to unify all these results, and although these additional results in themselves are not an improvement over existing results, the unifying proof can arguably be considered an improvement.

In chapter 8 we prove that it is graph isomorphism complete to check if a polytope, given by $\mathcal{HV}$-representation, is combinatorially isomorphic to its polar dual. This generalizes a previous result of Kaibel and Swartz [KS03] proving the GI-completeness of checking combinatorial isomorphism of two $\mathcal{HV}$-polytopes. Our result is based on a simple observation characterizing the decomposability of a polytope as free-join of smaller polytopes (Lemma 8.5.1).

9.1.2 New Hardness results

The most important new results in this thesis concern proving that many basic operations on polytopes like computing intersections, Minkowski sums, projection etc can not in general be performed in output-sensitive polynomial time unless $P = NP$. Although these results are negative, they do provide a positive view for algorithmic research on polytopes - it is very important to understand the output-sensitive complexity of vertex enumeration because many basic and practically important problems can only be efficiently solved via an intermediate step of converting the representation of the input polytopes to a certain form.

Many results in this thesis answer questions raised by others. In chapter 7 we derive many new hardness results related to the complexity of checking whether a given set of polyhedral cones cover a given set[1] D. As a corollary we answer a question of Bemporad et. al. [BFT01] in the negative by proving that it is NP-hard to decide whether the union of a given set of $\mathcal{H}$- or $\mathcal{V}$-polytopes is convex. Similarly, in chapter 8 we answer a question of Kaibel and Swartz [KS03] by proving that it is graph isomorphism complete to check whether a polytope, given by its facets and vertices, is combinatorially isomorphic to its polar dual.

Another novel contribution in chapter 8 is the construction of a family of self-dual polytopes via free-join. The polytopes constructed are interesting on their own because their incidence matrices are symmetrizable. In general transposability of the incidence matrix suffices for self-duality of a polytope and not all self-dual polytopes have symmetrizable incidence matrices. We also provide an example of a class of polytopes, which we call the roofed-prisms, that are self-dual but do not arise as a free-join of other polytopes. The observation that such polytopes exist is perhaps not surprising but roofed-prisms form an interesting class of such polytopes.

9.1.3 Completeness for VE

In this thesis we have defined a complexity class for problems that are equivalent to the vertex enumeration problem. This class tries to capture the notion of completeness for VE in the same way the class of NP-complete problems does for problems in NP. In chapter 6 the notion of a problem being equivalent, harder

[1]The sets that we explicitly consider are either the whole space or a linear subspace, although one can handle the case when D is a polytope with slight modifications.

or easier than VE is defined, and a class of problems related to computing the projection of polytopes is used as an example for these notions. It was shown how the various versions of the projection problem turn out to be either NP-hard, VE-complete, VE-hard or VE-easy.

Despite years of active research neither it is known whether VE is in P nor it is known to be NP-hard. This fact makes us believe that defining such a notion of completeness will turn out to be useful for future research. This thesis presents two problems that turn out to be related to this new complexity class (chapter 6, chapter 7) and we hope that many problems will be shown to have a complexity related to that of the vertex enumeration problem. Hopefully such new problems will make it possible to use tools from different areas for settling the question of the complexity of vertex enumeration.

9.1.4 Relating VE to GI

It is entirely possible that the complexity of vertex enumeration lies somewhere between P and NP-complete. We are not aware of any work trying to relate the complexity of vertex enumeration to some problem like graph isomorphism. Graph isomorphism, like vertex enumeration, has a long history of research and an unknown complexity status. Some results in this thesis try to take a step towards establishing the connection between these two problems. Even though we do not settle the question whether these two problems have similar complexities, we believe the results in chapter 8 are a step in the right direction.

In particular, we show in chapter 8 that a certain isomorphism problem defined for polytopes is graph isomorphism complete if and only if vertex enumeration is graph isomorphism easy. The problem, which we call SD or Self Duality, is shown to be vertex enumeration hard as well as graph isomorphism hard. As opposed to other problems considered in this thesis, that are shown to be NP-hard but whose NP-hardness offers no insight into the complexity of vertex enumeration, the self-duality problem will provide non-trivial insight into the complexity of vertex enumeration whether it turns out to be graph isomorphism complete or strictly harder that graph isomorphism (say NP-hard).

9.2 Future Work

The central problem that served as a motivation for this thesis - vertex enumeration - remains unsolved; neither do we have an output-sensitive algorithm for it, nor do we know whether it is NP-hard. Nevertheless, this thesis does offer a few insights into how research on this problem should be direction in future:

- Trying to work on a slightly more general problem usually offers no insight into the complexity of vertex enumeration. The new problems very frequently turn out to be not only harder than vertex enumeration but also NP-hard. It would be an interesting task to come up with problems that are more general than vertex enumeration but nevertheless not NP-hard. Naturally this assumes, without any strong evidence, that vertex enumeration is not NP-hard itself.
- The problem of checking self-duality of an $\mathcal{H}$- or $\mathcal{V}$-polytope shows some promise for at least partially settling the complexity of vertex enumeration. What is the complexity of this problem? Perhaps a key to understanding the complexity of this problem is to first understand the structure of self-dual polytopes. We constructed a nice class of self-dual polytopes in this thesis and it would be interesting to generate new classes of self-dual polytopes.
- Even the most basic operations of polytopes turn out to be either NP-hard or at least as hard as vertex enumeration in various representations. It would be of great help for practical applications if these operations can nevertheless be performed in reasonable time. It is perhaps important that when computing with polytopes we focus on the specific class of polytopes at hand and try to obtain a polynomial algorithm for that class.

List of Figures

List of Tables

Bibliography

[ABS97] David Avis, David Bremner, and Raimund Seidel. How good are convex hull algorithms. *Computational Geometry: Theory and Applications*, 7:265–302, 1997.

[ADP03] S. D. Abdullahi, M. E. Dyer, and L. G. Proll. Listing vertices of simple polyhedra associated with dual LI(2) systems. In *DMTCS: Discrete Mathematics and Theoretical Computer Science, 4th International Conference, DMTCS 2003, Proceedings*, pages 89–96, 2003.

[AF92] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8(3):295–313, 1992.

[AF96] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.

[AN93] Dan Archdeacon and Seiya Negami. The construction of self-dual projective polyhedra. *J. Comb. Theory, Ser. B*, 59(1):122–131, 1993.

[Bal88] Egon Balas. On the convex hull of the union of certain polyhedra. *Operations Research Letters*, 7:279–283, 1988.

[Bal98] Egon Balas. Projection with a minimal system of inequalities. *Comput. Optim. Appl.*, 10(2):189–193, 1998.

[BC06] Mark Braverman and Stephen Cook. Computing over the reals: Foundations for scientific computing. *Notices of the AMS*, 53(3):318–329, 2006.

[BEGM07] Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Generating vertices of polyhedra and related monotone generation problems. Technical Report 2007-03, DIMACS, Tue Apr 17 09:40:29 2007. Fri Apr 20 10:45:25 EDT 2007.

[BEGT08] Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Hans Raj Tiwary. The negative cycles polyhedron and hardness of checking some polyhedral properties. Technical report, Rutgers Center for Operations Research, 2008.

[BF05] Imre Bárány and Komei Fukuda. A case when the union of polytopes is convex. *Linear Algebra and its Applications*, 397(6):381–388, 2005.

[BFM98] David Bremner, Komei Fukuda, and Ambros Marzetta. Primal - dual methods for vertex and facet enumeration. *Discrete & Computational Geometry*, 20(3):333–357, 1998.

[BFT01] Alberto Bemporad, Komei Fukuda, and Fabio Danilo Torrisi. Convexity recognition of the union of polyhedra. *Comput. Geom*, 18(3):141–154, 2001.

[BGZ06] Endre Boros, Vladimir Gurvich, and Igor Zverovich. Neighborhood hypergraphs of bipartite graphs. Technical report, Rutgers Center for Operations Research, 2006.

[BHK04] A. Björklund, T. Husfeldt, and S. Khanna. Approximating longest directed paths and cycles. In *ICALP*, pages 222–233, 2004.

[BL98] Michael R. Bussieck and Marco E. Lübbecke. The vertex set of a 0/1 polytope is strongly $\mathcal{P}$-enumerable. *Computational Geometry: Theory and Applications*, 11(2):103–109, 1998.

[Cha93] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10:377–409, 1993.

[DF88] Martin E. Dyer and Alan M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.

[DFZ07] G. Ding, L. Feng, and W. Zang. The complexity of recognizing linear systems with certain integrality properties. *Math. Program., Ser. A., to appear*, 2007.

[Dye83] Martin E. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[EG95] Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.

[Eri00] Jeff Erickson. Faces of intricate polytopes. Notes on the Web: http://compgeom.cs.uiuc.edu/~jeffe/open/intricate.html, 2000.

[ET08] Khaled Elbassioni and Hans Raj Tiwary. On a cone covering problem. In *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG2008)*, pages 171–175, 2008.

[FLL01] Komei Fukuda, Thomas M. Liebling, and Christine Lutolf. Extended convex hull. *Computational Geometry*, 20(1-2):13–23, 2001.

[FO85] Robert M. Freund and James B. Orlin. On the complexity of four polyhedral set containment problems. *Mathematical Programming*, 33(2):139–145, 1985.

[Fuk04] Komei Fukuda. From the zonotope construction to the minkowski addition of convex polytopes. *J. Symb. Comput.*, 38(4):1261–1272, 2004.

[FW05a] Komei Fukuda and Christophe Weibel. Computing all faces of the minkowski sum of $\mathcal{V}$-polytopes. *In Proceedings of the 17th Canadian Conference on Computational Geometry*, 2005.

[FW05b] Komei Fukuda and Christophe Weibel. On f-vectors of minkowski additions of convex polytopes. *Technical report*, 2005. submitted to DCG.

[GLS93] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, 1993.

[Grü03] Branko Grünbaum. *Convex Polytopes Second Edition prepared by V. Kaibel, V. L. Klee and G. M. Ziegler*, volume 221 of *Graduate Texts in Mathematics*. Springer, 2003.

[GS88] Branko Grünbaum and Geoffrey C. Shephard. Is self-duality involutory? *American Mathematical Monthly*, 95:729–733, 1988.

[GS93] Peter Gritzmann and Bernd Sturmfels. Minkowski addition of polytopes: Computational complexity and applications to grobner bases. *SIJDM: SIAM Journal on Discrete Mathematics*, 6, 1993.

[GV95] N. Garg and V. V. Vazirani. A polyhedron with all s-t cuts as vertices, and adjacency of cuts. *Math. Program.*, 70(1):17–25, 1995.

[HRS00] Berkett Huber, Jörg Rambau, and Fransisco Santos. The cayley trick, lifting subdivisions and the bohne-dress theorem on zonotopal tilings. *Journal of the European Mathematical Society*, 2(2):179–198, 2000.

[Imb93] Jean-Louis Imbert. Fourier's elimination: Which to choose? In *PPCP*, pages 117–129, 1993.

[Jen89] Stanislav Jendrol. A non-involutory selfduality. *Discrete Mathematics*, 74(3):325–326, 1989.

[JKM04] C.N. Jones, E.C. Kerrigan, and J.M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical Report CUED/F-INFENG/TR.463, ETH Zurich, 2004.

[KBB$^+$06] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 758–765, 2006.

[Kha79] Leonid Khachian. A polynomial algorithm in linear programming. *Soviet Math. Dokl.*, 20(1):191–194, 1979.

[KLS98] Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $o^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11(1):1–50, December 1998.

[KP02] Volker Kaibel and Marc E. Pfetsch. Computing the face lattice of a polytope from its vertex-facet incidences. *Comput. Geom.*, 23(3):281–290, 2002.

[KS03] Volker Kaibel and Alexander Schwartz. On the complexity of polytope isomorphism problems. *Graphs and Combinatorics*, 19(2):215–230, 2003.

[Lin86] Nathan Linial. Hard enumeration problems in geometry and combinatorics. *SIAM J. Algebraic Discrete Methods*, 7(2):331–335, 1986.

[McM70] Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematica*, 17:179–184, 1970.

[MV99] Peter Bro Miltersen and N. Variyam Vinodchandran. Derandomizing arthur-merlin games using hitting sets. In *IEEE Symposium on Foundations of Computer Science*, pages 71–80, 1999.

[Pro94] J. S. Provan. Efficient enumeration of the vertices of polyhedra associated with network lp's. *Mathematical Programming*, 63(1):47–64, 1994.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[PY90] C. H. Papadimitriou and M. Yannakakis. On recognizing integer polyhedra. *Combinatorica*, 10(1):107–109, 1990.

[Rad07] Luis Rademacher. Approximating the centroid is hard. In *Symposium on Computational Geometry*, pages 302–305, 2007.

[RT75] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5:237–252, 1975.

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[Str94] Bernd Strumfels. On the newton polytope of the resultant. *Journal of Algebraic Combinatorics*, 3(2):207–236, 1994.

[TE08] Hans Raj Tiwary and Khaled M. Elbassioni. On the complexity of checking self-duality of polytopes and its relations to vertex enumeration and graph isomorphism. In *Symposium on Computational Geometry*, pages 192–198, 2008.

[Tiw07] Hans Raj Tiwary. On the hardness of minkowski addition and related operations. In *Symposium on Computational Geometry*, pages 306–309, 2007.

[Tiw08] Hans Raj Tiwary. On the hardness of computing intersection, union and minkowski sum of polytopes. *Discrete Comput. Geom.*, 40(3):469–479, 2008.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms.* Springer Verlag, Berlin, Heidelberg, New York, 2001.

[Yap00] Chee Keng Yap. *Fundamental Problems in Algorithmic Algebra.* Oxford University Press, New York, 2000.

[Zie95] Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer-Verlag, Berlin, 1995.