The complexity of optimizing over a simplex, hypercube or sphere

Etienne de Klerk

Tilburg University, The Netherlands

Generic optimization problem

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuous function and $K \subset \mathbb{R}^n$ a compact set.

Generic optimization problem

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuous function and $K \subset \mathbb{R}^n$ a compact set.

Generic optimization problems

$$\underline{f} := \min \{ f(x) \ : \ x \in K \},$$

Generic optimization problem

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuous function and $K \subset \mathbb{R}^n$ a compact set.

Generic optimization problems

$$\underline{f} := \min \{f(x) \ : \ x \in K\}, \quad \bar{f} := \max \{f(x) \ : \ x \in K\}.$$

Generic optimization problem

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuous function and $K \subset \mathbb{R}^n$ a compact set.

Generic optimization problems

$$\underline{f} := \min \{f(x) \ : \ x \in K\}, \quad \bar{f} := \max \{f(x) \ : \ x \in K\}.$$

Assumption:

We may compute $f(x)$ in time polynomial in the bit-size of x and in the bit size required to represent $f(x)$.

Choices for K

We consider the generic optimization problem for three choices of the compact set K:

Choices for K

We consider the generic optimization problem for three choices of the compact set K:

- the standard (or unit) simplex:

$$\Delta_n := \left\{ x \in \mathbb{R}^n \ : \ \sum_{i=1}^{n} x_i = 1, \ x \geq 0 \right\},$$

Choices for K

We consider the generic optimization problem for three choices of the compact set K:

- the standard (or unit) simplex:

$$\Delta_n := \left\{ x \in \mathbb{R}^n \ : \ \sum_{i=1}^{n} x_i = 1, \ x \geq 0 \right\},$$

- the unit hypercube $[0,1]^n$,

Choices for K

We consider the generic optimization problem for three choices of the compact set K:

- the standard (or unit) simplex:

$$\Delta_n := \left\{ x \in \mathbb{R}^n \ : \ \sum_{i=1}^{n} x_i = 1, \ x \geq 0 \right\},$$

- the unit hypercube $[0,1]^n$,
- the unit sphere: $S_n := \{x \in \mathbb{R}^n \ : \ \|x\| = 1\}$.

Example 1 for $K = \Delta_n$: maximum stable set

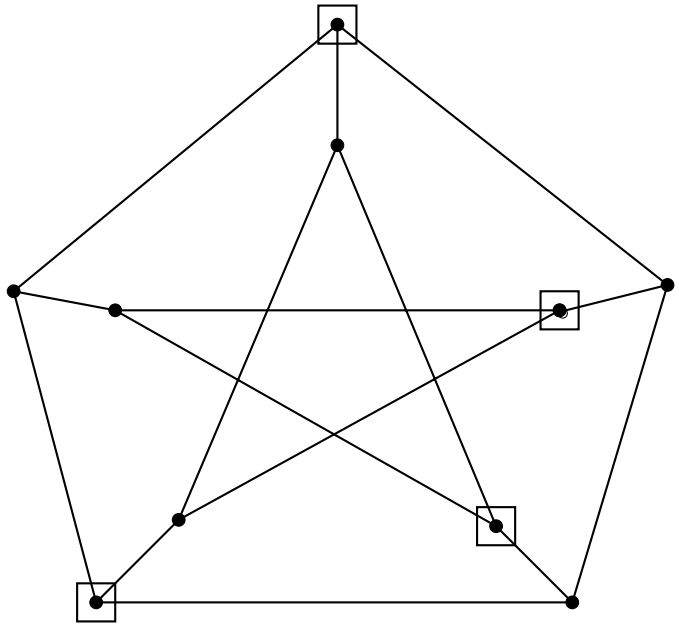
Maximum stable set problem

A subset of vertices $V' \subset V$ in a graph $G = (V, E)$ is called a *stable set* if no two vertices in V' are adjacent. The cardinality of the maximum stable set is denoted by $\alpha(G)$.

Example 1 for $K = \Delta_n$: maximum stable set

Maximum stable set problem

A subset of vertices $V' \subset V$ in a graph $G = (V, E)$ is called a *stable set* if no two vertices in V' are adjacent. The cardinality of the maximum stable set is denoted by $\alpha(G)$.



A maximum stable set of the Petersen graph

Example 1 for $K = \Delta_n$ (ctd.)

Theorem (Motzkin-Straus '65)

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta_{|V|}} x^T (A + I) x,$$

where A the adjacency matrix of the graph G and I the identity matrix.

Example 1 for $K = \Delta_n$ (ctd.)

Theorem (Motzkin-Straus '65)

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta_{|V|}} x^T (A + I) x,$$

where A the adjacency matrix of the graph G and I the identity matrix.

Consequence: Computing $\underline{f} := \min_{x \in \Delta_n} f(x)$ is NP-hard for any class of f that contains quadratic forms.

Example 2 for $K = \Delta_n$: computing the Lebesgue constant

Given a set of "poised" Lagrange interpolation points $\Theta \subset \Delta_n$.

Example 2 for $K = \Delta_n$: computing the Lebesgue constant

Given a set of "poised" Lagrange interpolation points $\Theta \subset \Delta_n$.

Denote the fundamental Lagrange polynomial associated with an interpolation point $\theta \in \Theta$ by ℓ_θ. In other words, for $\bar{\theta} \in \Theta$:

$$\ell_\theta(\bar{\theta}) = \begin{cases} 1 & \text{if } \bar{\theta} = \theta \\ 0 & \text{else.} \end{cases}$$

Example 2 for $K = \Delta_n$: computing the Lebesgue constant

Given a set of "poised" Lagrange interpolation points $\Theta \subset \Delta_n$.

Denote the fundamental Lagrange polynomial associated with an interpolation point $\theta \in \Theta$ by ℓ_θ. In other words, for $\bar{\theta} \in \Theta$:

$$\ell_\theta(\bar{\theta}) = \begin{cases} 1 & \text{if } \bar{\theta} = \theta \\ 0 & \text{else.} \end{cases}$$

The Lagrange interpolant (approximation) to a function g is then

$$L_\Theta(g)(x) := \sum_{\theta \in \Theta} g(\theta)\ell_\theta(x).$$

Example 2 for $K = \Delta_n$ (ctd.)

Definition (Lebesgue constant of Θ)

$$\Lambda(\Theta) := \max_{x \in \Delta_n} \sum_{\theta \in \Theta} |\ell_\theta(x)| \, .$$

Example 2 for $K = \Delta_n$ (ctd.)

Definition (Lebesgue constant of Θ)

$$\Lambda(\Theta) := \max_{x \in \Delta_n} \sum_{\theta \in \Theta} |\ell_\theta(x)| .$$

Theorem

Let p_g^* be the best polynomial approximation to g of the same degree as $L_\Theta(g)$ in the $\|\cdot\|_\infty$ norm.

Example 2 for $K = \Delta_n$ (ctd.)

Definition (Lebesgue constant of Θ)

$$\Lambda(\Theta) := \max_{x \in \Delta_n} \sum_{\theta \in \Theta} |\ell_\theta(x)| .$$

Theorem

Let p_g^* be the best polynomial approximation to g of the same degree as $L_\Theta(g)$ in the $\|\cdot\|_\infty$ norm. One has

$$\|L_\Theta(g) - g\|_\infty \leq (1 + \Lambda(\Theta))\|p_g^* - g\|_\infty.$$

Example 2 for $K = \Delta_n$ (ctd.)

Definition (Lebesgue constant of Θ)

$$\Lambda(\Theta) := \max_{x \in \Delta_n} \sum_{\theta \in \Theta} |\ell_\theta(x)| .$$

Theorem

Let p_g^* be the best polynomial approximation to g of the same degree as $L_\Theta(g)$ in the $\|\cdot\|_\infty$ norm. One has

$$\|L_\Theta(g) - g\|_\infty \leq (1 + \Lambda(\Theta))\|p_g^* - g\|_\infty.$$

Computing $\Lambda(\Theta)$ is an example of the generic optimization problem with $f(x) = \sum_{\theta \in \Theta} |\ell_\theta(x)|$ and $K = \Delta_n$.

Example for $K = S_n$ (unit sphere)

Let A be a real symmetric matrix with smallest eigenvalue $\lambda_{\min}(A)$.

Example for $K = S_n$ (unit sphere)

Let A be a real symmetric matrix with smallest eigenvalue $\lambda_{\min}(A)$.

Theorem (Raleigh-Ritz)

One has

$$\lambda_{\min}(A) = \min_{\|x\|=1} x^T A x.$$

Example for $K = S_n$ (unit sphere)

Let A be a real symmetric matrix with smallest eigenvalue $\lambda_{\min}(A)$.

Theorem (Raleigh-Ritz)

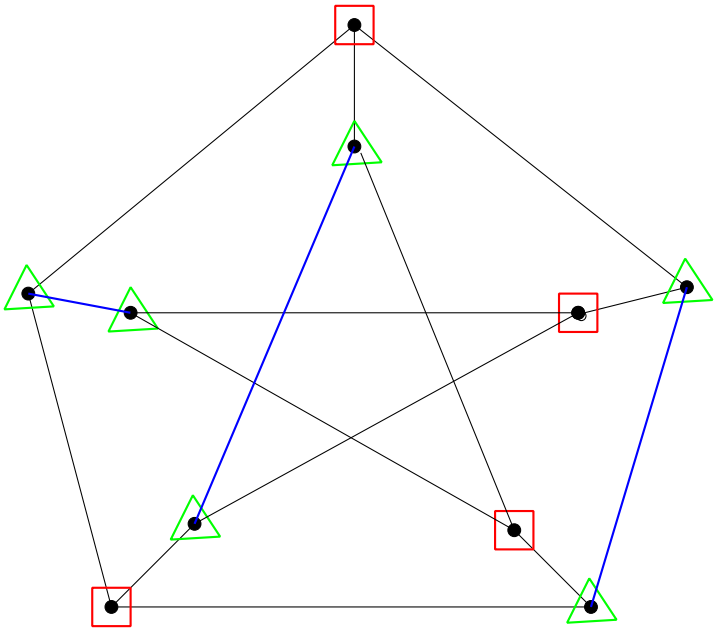
One has

$$\lambda_{\min}(A) = \min_{\|x\|=1} x^T A x.$$

Conclusion: optimizing quadratic forms over $K = S_n$ is 'easy' (unlike for $K = \Delta_n$).

Example for $K = [0, 1]^n$: the maximum cut problem

A maximum cut is a vertex colouring using two colours such that the number of edges with endpoints of different colours is maximal.



2–CUT of the Petersen graph

Example for $K = [0,1]^n$: the maximum cut problem (ctd.)

For a graph $G = (V, E)$ with Laplacian matrix L, the size of the maximum cut is given by:

$$|\text{maximum cut}| = \max_{x \in [-1,1]^{|V|}} \frac{1}{4} x^T L x$$

Example for $K = [0,1]^n$: the maximum cut problem (ctd.)

For a graph $G = (V, E)$ with Laplacian matrix L, the size of the maximum cut is given by:

$$|\text{maximum cut}| = \max_{x \in [-1,1]^{|V|}} \frac{1}{4} x^T L x = \max_{x \in [0,1]^{|V|}} \frac{1}{4} (2x - e)^T L (2x - e),$$

where e is the all-ones vector.

Example for $K = [0,1]^n$: the maximum cut problem (ctd.)

For a graph $G = (V, E)$ with Laplacian matrix L, the size of the maximum cut is given by:

$$|\text{maximum cut}| = \max_{x \in [-1,1]^{|V|}} \frac{1}{4} x^T L x = \max_{x \in [0,1]^{|V|}} \frac{1}{4} (2x - e)^T L (2x - e),$$

where e is the all-ones vector.

Consequence:

Optimizing quadratic polynomials over $[0,1]^n$ is NP-hard.

Outline of the rest of the talk

- Notions of approximation for continuous optimization;

Outline of the rest of the talk

- Notions of approximation for continuous optimization;
- Negative (in-approximabilily) results;

Outline of the rest of the talk

- Notions of approximation for continuous optimization;
- Negative (in-approximabilily) results;
- Approximation algorithms;

Outline of the rest of the talk

- Notions of approximation for continuous optimization;
- Negative (in-approximabilily) results;
- Approximation algorithms;
- Discussion.

ϵ-approximation

Definition ($(1-\epsilon)$-approximation)

A value ψ_ϵ is called a $(1-\epsilon)$-approximation of $\underline{f}$ for a given $\epsilon \in [0,1]$ if

$$|\psi_\epsilon - \underline{f}| \leq \epsilon(\bar{f} - \underline{f}).$$

ϵ-approximation

Definition ($(1-\epsilon)$-approximation)

A value ψ_ϵ is called a $(1-\epsilon)$-approximation of $\underline{f}$ for a given $\epsilon \in [0,1]$ if

$$|\psi_\epsilon - \underline{f}| \leq \epsilon(\bar{f} - \underline{f}).$$

The approximation is called *implementable* if $\psi_\epsilon = f(x_\epsilon)$ for some $x_\epsilon \in K$.

ϵ-approximation

Definition ($(1-\epsilon)$-approximation)

A value ψ_ϵ is called a $(1-\epsilon)$-approximation of $\underline{f}$ for a given $\epsilon \in [0,1]$ if

$$|\psi_\epsilon - \underline{f}| \leq \epsilon(\bar{f} - \underline{f}).$$

The approximation is called *implementable* if $\psi_\epsilon = f(x_\epsilon)$ for some $x_\epsilon \in K$.

Definition (Weak $(1-\epsilon)$-approximation)

If we use the condition

$$|\psi_\epsilon - \underline{f}| \leq \epsilon,$$

then we speak of a $(1-\epsilon)$-approximation of $\underline{f}$ in the weak sense.

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$.

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

1. For any $f \in \mathcal{F}$, $\mathcal{A}$ computes an $x_\epsilon \in K$ such that $f(x_\epsilon)$ is an implementable $(1-\epsilon)$-approximation of $\underline{f}$;

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

1. For any $f \in \mathcal{F}$, $\mathcal{A}$ computes an $x_\epsilon \in K$ such that $f(x_\epsilon)$ is an implementable $(1-\epsilon)$-approximation of $\underline{f}$;
2. the number of operations for computing x_ϵ is bounded by a polynomial in:

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

1. For any $f \in \mathcal{F}$, $\mathcal{A}$ computes an $x_\epsilon \in K$ such that $f(x_\epsilon)$ is an implementable $(1-\epsilon)$-approximation of $\underline{f}$;
2. the number of operations for computing x_ϵ is bounded by a polynomial in:
   (i) The number of variables n;

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

1. For any $f \in \mathcal{F}$, $\mathcal{A}$ computes an $x_\epsilon \in K$ such that $f(x_\epsilon)$ is an implementable $(1-\epsilon)$-approximation of $\underline{f}$;
2. the number of operations for computing x_ϵ is bounded by a polynomial in:
   (i) The number of variables n;
   (ii) the bit size required to represent f.

Polynomial time approximation algorithm

Definition (Polynomial time approximation algorithm)

Fix $\epsilon > 0$ and a class of continuous, computable functions on K, say $\mathcal{F}$. An algorithm $\mathcal{A}$ is called a polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$, if:

1. For any $f \in \mathcal{F}$, $\mathcal{A}$ computes an $x_\epsilon \in K$ such that $f(x_\epsilon)$ is an implementable $(1-\epsilon)$-approximation of $\underline{f}$;
2. the number of operations for computing x_ϵ is bounded by a polynomial in:
   (i) The number of variables n;
   (ii) the bit size required to represent f.

If the number of operations to compute x_ϵ is also bounded by a polynomial in $1/\epsilon$, then $\mathcal{A}$ is called a *strongly* polynomial time $(1-\epsilon)$-approximation algorithm for $\mathcal{F}$.

PTAS/FPTAS

Definition (PTAS/FPTAS)

- If, for a given function class $\mathcal{F}$, there exists a polynomial-time $(1-\epsilon)$-approximation algorithm for each $\epsilon \in (0, 1]$, we say that there is a polynomial time approximation scheme (PTAS) for $\mathcal{F}$.

PTAS/FPTAS

Definition (PTAS/FPTAS)

- If, for a given function class $\mathcal{F}$, there exists a polynomial-time $(1-\epsilon)$-approximation algorithm for each $\epsilon \in (0,1]$, we say that there is a polynomial time approximation scheme (PTAS) for $\mathcal{F}$.
- In case of a strongly polynomial time $(1-\epsilon)$-approximation algorithm for each $\epsilon \in (0,1]$, we speak of a fully polynomial time approximation scheme (FPTAS).

PTAS/FPTAS

Definition (PTAS/FPTAS)

- If, for a given function class $\mathcal{F}$, there exists a polynomial-time $(1-\epsilon)$-approximation algorithm for each $\epsilon \in (0,1]$, we say that there is a polynomial time approximation scheme (PTAS) for $\mathcal{F}$.
- In case of a strongly polynomial time $(1-\epsilon)$-approximation algorithm for each $\epsilon \in (0,1]$, we speak of a fully polynomial time approximation scheme (FPTAS).

These definitions can be adapted in an obvious way for maximization problems, or for approximations are in the weak sense.

Negative results for Δ_n

Recall that $\alpha(G)$ denotes the stability number of a graph G, and

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta_{|V|}} x^T (A + I) x,$$

Negative results for Δ_n

Recall that $\alpha(G)$ denotes the stability number of a graph G, and

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta_{|V|}} x^T (A + I)x,$$

Theorem (Håstad)

Unless NP=ZPP, one cannot approximate $\alpha(G)$ to within a factor $|V|^{(1-\epsilon)}$ in polynomial time for any $\epsilon > 0$.

Negative results for Δ_n

Recall that $\alpha(G)$ denotes the stability number of a graph G, and

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta_{|V|}} x^T(A+I)x,$$

Theorem (Håstad)

Unless NP=ZPP, one cannot approximate $\alpha(G)$ to within a factor $|V|^{(1-\epsilon)}$ in polynomial time for any $\epsilon > 0$.

Corollary

Unless NP=ZPP, there is no FPTAS for optimizing the class of quadratic functions over $K = \Delta_n$.

Negative results for $[0,1]^n$

Recall

$$|\text{maximum cut}| = \max_{x\in[-1,1]^{|V|}} \frac{1}{4}x^T Lx = \max_{x\in[0,1]^{|V|}} \frac{1}{4}(2x-e)^T L(2x-e),$$

where L is the Laplacian matrix of a graph G.

Negative results for $[0,1]^n$

Recall

$$|\text{maximum cut}| = \max_{x\in[-1,1]^{|V|}} \frac{1}{4}x^T Lx = \max_{x\in[0,1]^{|V|}} \frac{1}{4}(2x-e)^T L(2x-e),$$

where L is the Laplacian matrix of a graph G.

Theorem (Håstad)

Unless P=NP, these is no polynomial time $16/17$*-approximation algorithm for the maximum cut problem.*

Negative results for $[0,1]^n$

Recall

$$|\text{maximum cut}| = \max_{x\in[-1,1]^{|V|}} \frac{1}{4}x^T Lx = \max_{x\in[0,1]^{|V|}} \frac{1}{4}(2x-e)^T L(2x-e),$$

where L is the Laplacian matrix of a graph G.

Theorem (Håstad)

Unless P=NP, these is no polynomial time $16/17$*-approximation algorithm for the maximum cut problem.*

Corollary

There is no PTAS for optimizing any class of functions that includes the quadratic polynomials over $K = [0,1]^n$.

Negative results for the unit sphere

Theorem (Nesterov)

Consider a graph $G = (V, E)$ with stability number $\alpha(G)$. One has

$$\sqrt{1 - \frac{1}{\alpha(G)}} = 3\sqrt{3} \max_{\|x\|^2+\|y\|^2=1} \sum_{\substack{i<j \\ \{i,j\}\notin E}} y_{ij} x_i x_j.$$

Negative results for the unit sphere

Theorem (Nesterov)

Consider a graph $G = (V, E)$ with stability number $\alpha(G)$. One has

$$\sqrt{1 - \frac{1}{\alpha(G)}} = 3\sqrt{3} \max_{\|x\|^2 + \|y\|^2 = 1} \sum_{\substack{i<j \\ \{i,j\} \notin E}} y_{ij} x_i x_j.$$

In view of the inapproximability result for the maximum stable set problem, we have:

Negative results for the unit sphere

Theorem (Nesterov)

Consider a graph $G = (V, E)$ with stability number $\alpha(G)$. One has

$$\sqrt{1 - \frac{1}{\alpha(G)}} = 3\sqrt{3} \max_{\|x\|^2+\|y\|^2=1} \sum_{\substack{i<j \\ \{i,j\}\notin E}} y_{ij} x_i x_j.$$

In view of the inapproximability result for the maximum stable set problem, we have:

Corollary

Unless NP=ZPP, there is no FPTAS for minimizing any class of functions that includes square free degree 3 forms over the unit sphere.

Approximation results for Δ_n

A simple approximation algorithm

Define the following rational grid on Δ_n,

$$\Delta(n,m) := \{x \in \Delta_n \ : \ mx \in \mathbb{N}_0^n\},$$

Approximation results for Δ_n

A simple approximation algorithm

Define the following rational grid on Δ_n,

$$\Delta(n,m) := \{x \in \Delta_n \ : \ mx \in \mathbb{N}_0^n\},$$

and compute the value

$$f_{\Delta(n,m)} := \min_{x \in \Delta(n,m)} f(x).$$

Approximation results for Δ_n

A simple approximation algorithm

Define the following rational grid on Δ_n,

$$\Delta(n,m) := \{x \in \Delta_n \ : \ mx \in \mathbb{N}_0^n\},$$

and compute the value

$$f_{\Delta(n,m)} := \min_{x \in \Delta(n,m)} f(x).$$

Note that $|\Delta(n,m)| = \binom{n+m}{m}$ which is a polynomial in n for fixed m.

Approximation results for Δ_n (ctd.)

Theorem (Bomze, De Klerk)

Let f be quadratic. One has

$$f_{\Delta(n,m)} - \underline{f} \leq \frac{1}{m}(\bar{f} - \underline{f})$$

for any $m \geq 1$.

Approximation results for Δ_n (ctd.)

Theorem (Bomze, De Klerk)

Let f be quadratic. One has

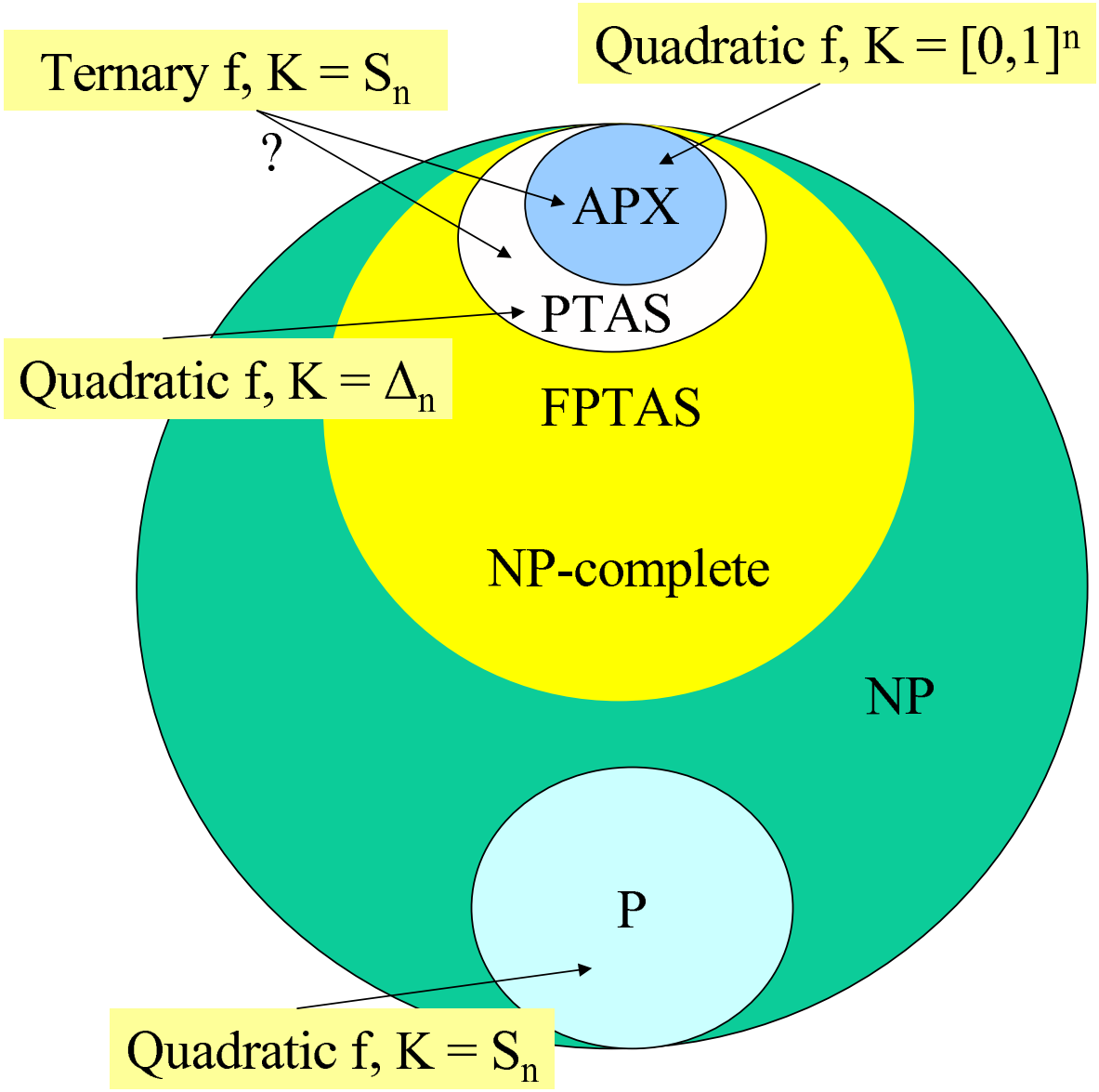
$$f_{\Delta(n,m)} - \underline{f} \leq \frac{1}{m}(\bar{f} - \underline{f})$$

for any $m \geq 1$.

Corollary (Bomze, De Klerk)

There exists a PTAS for minimizing quadratic polynomials over the unit simplex.

Summary so far



Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Laurent, and Parrilo)

Let $f(x)$ be a form of degree d and $r \geq 0$ an integer.

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Laurent, and Parrilo)

Let $f(x)$ be a form of degree d and $r \geq 0$ an integer. Then,

$$f_{\Delta(n,r+d)} - \underline{f} \leq (1 - w_r(d)) \binom{2d-1}{d} d^d (\bar{f} - \underline{f}),$$

where

$$w_r(d) := \frac{(r+d)!}{r!(r+d)^d} = \prod_{i=1}^{d-1} \left(1 - \frac{i}{r+d}\right).$$

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Laurent, and Parrilo)

Let $f(x)$ be a form of degree d and $r \geq 0$ an integer. Then,

$$f_{\Delta(n,r+d)} - \underline{f} \leq (1 - w_r(d)) \binom{2d-1}{d} d^d (\bar{f} - \underline{f}),$$

where

$$w_r(d) := \frac{(r+d)!}{r!(r+d)^d} = \prod_{i=1}^{d-1} \left(1 - \frac{i}{r+d}\right).$$

One can verify that $\lim_{r\to\infty} w_r(d) = 1$.

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Laurent, and Parrilo)

Let $f(x)$ be a form of degree d and $r \geq 0$ an integer. Then,

$$f_{\Delta(n,r+d)} - \underline{f} \leq (1 - w_r(d)) \binom{2d-1}{d} d^d (\bar{f} - \underline{f}),$$

where

$$w_r(d) := \frac{(r+d)!}{r!(r+d)^d} = \prod_{i=1}^{d-1} \left(1 - \frac{i}{r+d}\right).$$

One can verify that $\lim_{r\to\infty} w_r(d) = 1$.

Corollary (De Klerk, Laurent, and Parrilo)

Fix $d \in \mathbb{N}$. There exists a PTAS for minimizing forms of fixed degree d over the unit simplex.

Approximation results for Δ_n (ctd.)

We now consider more general f than polynomials.

Approximation results for Δ_n (ctd.)

We now consider more general f than polynomials.

Definition (Modulus of continuity)

The modulus of continuity of f on a compact convex set K is defined by

$$\omega(f,\delta) := \max_{\substack{x,y\in K \\ \|x-y\|\leq\delta}} |f(x)-f(y)|.$$

Approximation results for Δ_n (ctd.)

We now consider more general f than polynomials.

Definition (Modulus of continuity)

The modulus of continuity of f on a compact convex set K is defined by

$$\omega(f,\delta) := \max_{\substack{x,y\in K \\ \|x-y\|\leq\delta}} |f(x)-f(y)|.$$

Definition (Hölder continuity)

The class of Hölder continuous f:

$$\mathrm{Lip}_L(\alpha) := \{f \in C(\Delta_n) \ : \ \omega(f,\delta) \leq \delta^\alpha L \quad \forall\, \delta > 0\}\,.$$

Approximation results for Δ_n (ctd.)

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Elabwabi, Den Hertog)

Let $\epsilon > 0$, $\alpha > 0$, and $L > 0$ be given and assume $f \in Lip_L(\alpha)$.

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Elabwabi, Den Hertog)

Let $\epsilon > 0$, $\alpha > 0$, and $L > 0$ be given and assume $f \in Lip_L(\alpha)$. Then, for

$$m = \left\lceil \left(\frac{2L}{\epsilon} \right)^{\frac{2}{\alpha}} \right\rceil,$$

one has

$$f_{\Delta(n,m)} - \underline{f} \leq \epsilon.$$

Approximation results for Δ_n (ctd.)

Theorem (De Klerk, Elabwabi, Den Hertog)

Let $\epsilon > 0$, $\alpha > 0$, and $L > 0$ be given and assume $f \in Lip_L(\alpha)$. Then, for

$$m = \left\lceil \left(\frac{2L}{\epsilon} \right)^{\frac{2}{\alpha}} \right\rceil,$$

one has

$$f_{\Delta(n,m)} - \underline{f} \leq \epsilon.$$

Corollary

There is a PTAS in the weak sense for minimizing computable functions from the class $Lip_L(\alpha)$ over Δ_n, for fixed L and α.

Approximation results for $[0,1]^n$

Theorem (Nesterov)

There exists a (randomized) polynomial time $2/\pi$ approximation algorithm for the problem of maximizing a convex quadratic function over $[0,1]^n$.

Approximation results for $[0,1]^n$

Theorem (Nesterov)

There exists a (randomized) polynomial time $2/\pi$ approximation algorithm for the problem of maximizing a convex quadratic function over $[0,1]^n$.

Special case: maximum cut

For the special case of the maximum cut problem there is a 0.878 approximation algorithm due to Goemans and Williamson.

Approximation results for $[0,1]^n$

Theorem (Nesterov)

There exists a (randomized) polynomial time $2/\pi$ approximation algorithm for the problem of maximizing a convex quadratic function over $[0,1]^n$.

Special case: maximum cut

For the special case of the maximum cut problem there is a 0.878 approximation algorithm due to Goemans and Williamson.

The underlying approximation algorithms for these problems use semidefinite programming.

Approximation results for the unit sphere

The results for Δ_n imply:

Theorem (De Klerk, Laurent, Parrilo)

There is a PTAS for optimizing even forms (all exponents even) of fixed degree over the unit sphere.

Approximation results for the unit sphere

The results for Δ_n imply:

Theorem (De Klerk, Laurent, Parrilo)

There is a PTAS for optimizing even forms (all exponents even) of fixed degree over the unit sphere.

Another recent result:

Theorem (Barvinok)

There is a (randomized) PTAS for optimizing the class of so-called (δ, N)-focused forms of fixed degree over the unit sphere.

Approximation results for the unit sphere

The results for Δ_n imply:

Theorem (De Klerk, Laurent, Parrilo)

There is a PTAS for optimizing even forms (all exponents even) of fixed degree over the unit sphere.

Another recent result:

Theorem (Barvinok)

There is a (randomized) PTAS for optimizing the class of so-called (δ, N)-focused forms of fixed degree over the unit sphere.

It is an open question whether there is a PTAS for all forms of fixed degree over S_n.

Summary

Complexity of computing

$$\underline{f} := \min \{f(x) \ : \ x \in K\}$$

for the three choices of K and different classes of f.

	Δ_n	S_n	$[0,1]^n$
f quadratic	PTAS, no FPTAS	in P	no PTAS, $2/\pi$-approximation if f concave
f degree 3	PTAS	no FPTAS	no PTAS
f fixed degree	PTAS	PTAS for even or focused forms	no PTAS
$f \in \text{Lip}_L(\alpha)$	weak PTAS	?	no PTAS

Conclusion/discussion

- We have seen that optimization over the hypercube is much harder than over the simplex, while the complexity of optimization over a sphere is somewhere in between.

Conclusion/discussion

- We have seen that optimization over the hypercube is much harder than over the simplex, while the complexity of optimization over a sphere is somewhere in between.
- Approximation algorithms have been studied extensively for combinatorial optimization problems, but not for NP-hard continuous optimization problems.

Conclusion/discussion

- We have seen that optimization over the hypercube is much harder than over the simplex, while the complexity of optimization over a sphere is somewhere in between.
- Approximation algorithms have been studied extensively for combinatorial optimization problems, but not for NP-hard continuous optimization problems.
- Not much computational experience with approximation algorithms for continuous optimization either.

Conclusion/discussion

- We have seen that optimization over the hypercube is much harder than over the simplex, while the complexity of optimization over a sphere is somewhere in between.
- Approximation algorithms have been studied extensively for combinatorial optimization problems, but not for NP-hard continuous optimization problems.
- Not much computational experience with approximation algorithms for continuous optimization either.
- More theoretical and computational research needed in this area!

Conclusion/discussion

- We have seen that optimization over the hypercube is much harder than over the simplex, while the complexity of optimization over a sphere is somewhere in between.
- Approximation algorithms have been studied extensively for combinatorial optimization problems, but not for NP-hard continuous optimization problems.
- Not much computational experience with approximation algorithms for continuous optimization either.
- More theoretical and computational research needed in this area! Any interest?