Enumeration of the monomials of a polynomial

Yann Strozecki

Université Paris Diderot - Paris 7

Dept. of Computer Science, University of Toronto

January 2011, Toronto
Theory Seminar

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)
- $\sharp\{y | A(x, y)\}$: counting problem (class $\sharp\mathrm{P}$)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)
- $\sharp\{y | A(x, y)\}$: counting problem (class $\sharp\mathrm{P}$)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.
- The counting problem is to count the number of perfect matchings.

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)
- $\sharp\{y | A(x, y)\}$: counting problem (class $\sharp\mathrm{P}$)
- $\{y | A(x, y)\}$: enumeration problem (class EnumP)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.
- The counting problem is to count the number of perfect matchings.

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)
- $\sharp\{y | A(x, y)\}$: counting problem (class $\sharp\mathrm{P}$)
- $\{y | A(x, y)\}$: enumeration problem (class EnumP)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.
- The counting problem is to count the number of perfect matchings.
- The enumeration problem is to find every perfect matching.

Enumeration problems

Polynomially balanced predicate $A(x, y)$, decidable in polynomial time.

- $\exists ? y A(x, y)$: decision problem (class NP)
- $\sharp\{y | A(x, y)\}$: counting problem (class $\sharp\mathrm{P}$)
- $\{y | A(x, y)\}$: enumeration problem (class EnumP)

Example

The predicate $A(x, y)$ means y is a perfect matching in the graph x.

- The decision problem is to decide if there is a perfect matching.
- The counting problem is to count the number of perfect matchings.
- The enumeration problem is to find every perfect matching.

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: **TotalP**

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: **TotalP**
2. the delay

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: **TotalP**
2. the delay
   - incremental polynomial time: **IncP**

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: **TotalP**
2. the delay
   - incremental polynomial time: **IncP**
   - polynomial delay: **DelayP** (Perfect Matching [Uno])

Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: $\mathbf{TotalP}$
2. the delay
   - incremental polynomial time: $\mathbf{IncP}$
   - polynomial delay: $\mathbf{DelayP}$ (Perfect Matching [Uno])

Open question: is $\mathbf{DelayP} \neq \mathbf{IncP}$ modulo some complexity hypothesis ?

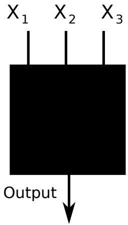
Complexity measures for enumeration

For enumeration problems we have two interesting complexity measures:

1. the total time related to the number of solutions
   - polynomial total time: $\mathbf{TotalP}$
2. the delay
   - incremental polynomial time: $\mathbf{IncP}$
   - polynomial delay: $\mathbf{DelayP}$ (Perfect Matching [Uno])

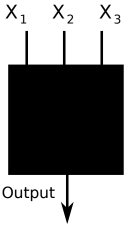
Open question: is $\mathbf{DelayP} \neq \mathbf{IncP}$ modulo some complexity hypothesis ?

Polynomial given by a black-box



$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

Polynomial given by a black-box



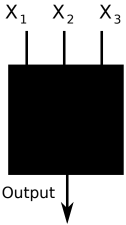
$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

$$X_1 = 1,\ X_2 = 2,\ X_3 = 1$$

$$1 * 2 + 1 * 1 + 2 + 1$$

$$Output = 6$$

Polynomial given by a black-box



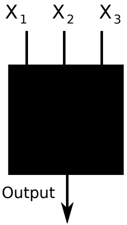
$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

$$X_1 = -1,\ X_2 = 1,\ X_3 = 2$$

$$-1 * 1 + -1 * 2 + 1 + 2$$

$$Output = 0$$

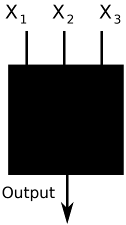
Polynomial given by a black-box



$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

- Problem: interpolation, compute P from its values.
- Parameters: number of variables and total degree.

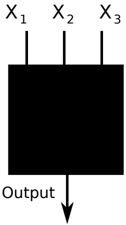
Polynomial given by a black-box



$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

- Problem: interpolation, compute P from its values.
- Parameters: number of variables and total degree.
- Complexity: time and number of calls to the oracle.

Polynomial given by a black-box

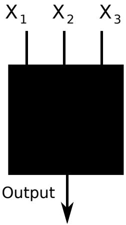


$$P(X_1, X_2, X_3) = X_1 X_2 + X_1 X_3 + X_2 + X_3$$

- Problem: interpolation, compute P from its values.
- Parameters: number of variables and total degree.
- Complexity: time and number of calls to the oracle.

Enumeration problem: output the monomials one after the other.

Polynomial given by a black-box



$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$$

- Problem: interpolation, compute P from its values.
- Parameters: number of variables and total degree.
- Complexity: time and number of calls to the oracle.

Enumeration problem: output the monomials one after the other.

Existing interpolation methods

- Zippel (1990): use a dense interpolation on a polynomial with a restricted number of variables
- Ben Or and Tiwari (1988): evaluation on big power of prime numbers

Existing interpolation methods

- Zippel (1990): use a dense interpolation on a polynomial with a restricted number of variables
- Ben Or and Tiwari (1988): evaluation on big power of prime numbers
- Klivans and Spielman (2001): transformation of a multivariate into an univariate one.

Existing interpolation methods

- Zippel (1990): use a dense interpolation on a polynomial with a restricted number of variables
- Ben Or and Tiwari (1988): evaluation on big power of prime numbers
- Klivans and Spielman (2001): transformation of a multivariate into an univariate one.
- Garg and Schost (2009): non black-box but complexity independent from the degree of the polynomial

Existing interpolation methods

- Zippel (1990): use a dense interpolation on a polynomial with a restricted number of variables
- Ben Or and Tiwari (1988): evaluation on big power of prime numbers
- Klivans and Spielman (2001): transformation of a multivariate into an univariate one.
- Garg and Schost (2009): non black-box but complexity independent from the degree of the polynomial

Enumeration complexity: produce the monomials one at a time with a good **delay**.

Existing interpolation methods

- Zippel (1990): use a dense interpolation on a polynomial with a restricted number of variables
- Ben Or and Tiwari (1988): evaluation on big power of prime numbers
- Klivans and Spielman (2001): transformation of a multivariate into an univariate one.
- Garg and Schost (2009): non black-box but complexity independent from the degree of the polynomial

Enumeration complexity: produce the monomials one at a time with a good **delay**.

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph
- Determinant of the Kirchoff matrix: spanning trees

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph
- Determinant of the Kirchoff matrix: spanning trees
- Determinant of the Tutte matrix: perfect matchings of bipartite graphs

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph
- Determinant of the Kirchoff matrix: spanning trees
- Determinant of the Tutte matrix: perfect matchings of bipartite graphs
- Pfaffian Hypertree theorem [Masbaum and Vaintraub 2002]: spanning hypertrees of a 3-uniform hypergraph

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph
- Determinant of the Kirchoff matrix: spanning trees
- Determinant of the Tutte matrix: perfect matchings of bipartite graphs
- Pfaffian Hypertree theorem [Masbaum and Vaintraub 2002]: spanning hypertrees of a 3-uniform hypergraph

Only multilinear polynomials.

Motivation

Easy to evaluate polynomials whose monomials represent interesting combinatorial objects.

- Determinant of the adjacency matrix : cycle covers of a graph
- Determinant of the Kirchoff matrix: spanning trees
- Determinant of the Tutte matrix: perfect matchings of bipartite graphs
- Pfaffian Hypertree theorem [Masbaum and Vaintraub 2002]: spanning hypertrees of a 3-uniform hypergraph

Only multilinear polynomials.

The decision problem

Polynomial Identity Testing
Input: a polynomial given as a black box.
Output: decides if the polynomial is zero.

The decision problem

POLYNOMIAL IDENTITY TESTING
Input: a polynomial given as a black box.
Output: decides if the polynomial is zero.

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if $x_1, \ldots, x_n$ are randomly chosen in a set of integers S of size $\frac{D}{\epsilon}$ then the probability that $P(x_1, \ldots, x_n) = 0$ is bounded by ϵ.

The decision problem

Polynomial Identity Testing
Input: a polynomial given as a black box.
Output: decides if the polynomial is zero.

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if $x_1, \ldots, x_n$ are randomly chosen in a set of integers S of size $\frac{D}{\epsilon}$ then the probability that $P(x_1, \ldots, x_n) = 0$ is bounded by ϵ.

No way to make PIT deterministic for black box.

The decision problem

Polynomial Identity Testing
Input: a polynomial given as a black box.
Output: decides if the polynomial is zero.

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if $x_1, \ldots, x_n$ are randomly chosen in a set of integers S of size $\frac{D}{\epsilon}$ then the probability that $P(x_1, \ldots, x_n) = 0$ is bounded by ϵ.

No way to make PIT deterministic for black box.

Error **exponentially small** in the size of the integers!

The decision problem

POLYNOMIAL IDENTITY TESTING
Input: a polynomial given as a black box.
Output: decides if the polynomial is zero.

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if $x_1, \ldots, x_n$ are randomly chosen in a set of integers S of size $\frac{D}{\epsilon}$ then the probability that $P(x_1, \ldots, x_n) = 0$ is bounded by ϵ.

No way to make PIT deterministic for black box.

Error **exponentially small** in the size of the integers!

From finding a monomial to interpolation

Assume there is a procedure which returns a monomial of a polynomial P, then it can be used to interpolate P.

Idea: Substract the monomial found by the procedure to the polynomial and recurse to recover the whole polynomial.

From finding a monomial to interpolation

Assume there is a procedure which returns a monomial of a polynomial P, then it can be used to interpolate P.

Idea: Substract the monomial found by the procedure to the polynomial and recurse to recover the whole polynomial.

Drawback: one has to store the polynomial which is the sum of the generated monomials and to evaluate it at each step.

From finding a monomial to interpolation

Assume there is a procedure which returns a monomial of a polynomial P, then it can be used to interpolate P.

Idea: Substract the monomial found by the procedure to the polynomial and recurse to recover the whole polynomial.

Drawback: one has to store the polynomial which is the sum of the generated monomials and to evaluate it at each step.

Incremental time.

From finding a monomial to interpolation

Assume there is a procedure which returns a monomial of a polynomial P, then it can be used to interpolate P.

Idea: Substract the monomial found by the procedure to the polynomial and recurse to recover the whole polynomial.

Drawback: one has to store the polynomial which is the sum of the generated monomials and to evaluate it at each step.

Incremental time.

Polynomials whose monomials have distinct supports

The *support* of a monomial is the set of indices of variables which appears in the monomial. The support of $X_1 X_3^2 X_5$ is $\{1, 3, 5\}$.

Write P_L for the polynomial P where all variables with indices outside of L set to 0.

Example

$$P = X_1 X_3^2 X_5 + X_2^4 X_3 + X_1 X_4 + X_2$$

$$P_{\{2,3,4\}} = X_2^4 X_3 + X_2$$

Polynomials whose monomials have distinct supports

The *support* of a monomial is the set of indices of variables which appears in the monomial. The support of $X_1 X_3^2 X_5$ is $\{1, 3, 5\}$.

Write P_L for the polynomial P where all variables with indices outside of L set to 0.

Example

$$P = X_1 X_3^2 X_5 + X_2^4 X_3 + X_1 X_4 + X_2$$

$$P_{\{2,3,4\}} = X_2^4 X_3 + X_2$$

Lemma

Let P be a polynomial without constant term and whose monomials have different supports and L a minimal set (for inclusion) of variables such that P_L is not identically zero. Then P_L is a monomial of support L.

Polynomials whose monomials have distinct supports

The *support* of a monomial is the set of indices of variables which appears in the monomial. The support of $X_1X_3^2X_5$ is $\{1,3,5\}$.

Write P_L for the polynomial P where all variables with indices outside of L set to 0.

Example

$$P = X_1X_3^2X_5 + X_2^4X_3 + X_1X_4 + X_2$$

$$P_{\{2,3,4\}} = X_2^4X_3 + X_2$$

Lemma

Let P be a polynomial without constant term and whose monomials have different supports and L a minimal set (for inclusion) of variables such that P_L is not identically zero. Then P_L is a monomial of support L.

Finding one monomial

Hypothesis: polynomials whose monomials have distinct supports

Easy to find a monomial of minimal support with a polynomial number of calls to the black box

Finding one monomial

Hypothesis: polynomials whose monomials have distinct supports

Easy to find a monomial of minimal support with a polynomial number of calls to the black box

- build a minimal set L such that P_L is not zero by successively setting each variable to 0 while the polynomial is not zero (property verified by a probabilistic test)

Finding one monomial

Hypothesis: polynomials whose monomials have distinct supports

Easy to find a monomial of minimal support with a polynomial number of calls to the black box

- build a minimal set L such that P_L is not zero by successively setting each variable to 0 while the polynomial is not zero (property verified by a probabilistic test)
- once we have found the support determine the degree and coefficient by some appropriate evaluation of P_L

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,\ X_4 = 1$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,$ $X_4 = 1$
  $\to 1$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,$ $X_4 = 1$ $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2,$ $X_4 = 3$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1$, $X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2$, $X_4 = 3$
  $\rightarrow 4$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,\ X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2,\ X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1$, $X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2$, $X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,\ X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2,\ X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$
- $L = \{3\}$ evaluation of P_L on $X_3 = 1$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1$, $X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2$, $X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$
- $L = \{3\}$ evaluation of P_L on $X_3 = 1$
  $\rightarrow 1$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,$ $X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2,$ $X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$
- $L = \{3\}$ evaluation of P_L on $X_3 = 1$
  $\rightarrow 1$
- stop

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1$, $X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2$, $X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$
- $L = \{3\}$ evaluation of P_L on $X_3 = 1$
  $\rightarrow 1$
- stop

Support: $L = \{3\}$

Building L, an example:

$$P(X_1, X_2, X_3, X_4) = X_1^3 X_2 + X_1 X_3 - 3X_2 X_4 + X_3^2$$

$L = \{1, 2, 3, 4\}$

- $L = \{2, 3, 4\}$ evaluation of P_L on $X_2 = 1, X_3 = 1,\ X_4 = 1$
  $\rightarrow 1$
- $L = \{3, 4\}$ evaluation of P_L on $X_3 = 2,\ X_4 = 3$
  $\rightarrow 4$
- $L = \{4\}$ evaluation of P_L on $X_4 = 2$
  $\rightarrow 0$
- $L = \{3\}$ evaluation of P_L on $X_3 = 1$
  $\rightarrow 1$
- stop

Support: $L = \{3\}$

Finding one monomial

Hypothesis: polynomials whose monomials have distinct supports

Easy to find a monomial of minimal support with a polynomial number of calls to the black box

- build a minimal set L such that P_L is not zero by successively setting each variable to 0 while the polynomial is not zero (property verified by a probabilistic test)
- once we have found the support determine the degree and coefficient by some appropriate evaluation of P_L

This procedure allows to find a monomial in **polynomial time** in the number of variables and the degree and with a probability of error **exponentially** small.

Finding one monomial

Hypothesis: polynomials whose monomials have distinct supports

Easy to find a monomial of minimal support with a polynomial number of calls to the black box

- build a minimal set L such that P_L is not zero by successively setting each variable to 0 while the polynomial is not zero (property verified by a probabilistic test)
- once we have found the support determine the degree and coefficient by some appropriate evaluation of P_L

This procedure allows to find a monomial in **polynomial time** in the number of variables and the degree and with a probability of error **exponentially** small.

Theorem

Let P be a polynomial whose monomials have distinct supports with n variables, t monomials and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1-\epsilon$. The delay between the i^{th} and $i+1^{th}$ monomials is bounded by $O(iDn^2(n+\log(\epsilon^{-1})))$ in time and $O(n(n+\log(\epsilon^{-1})))$ calls to the oracle. The algorithm performs $O(tn(n+\log(\epsilon^{-1})))$ calls to the oracle on points of size $\log(2D)$.

Delay: incremental in time and polynomial in the number of calls to the oracle.

Partial degree

We want to determine the degree of a subset S of variables of the polynomial.

1. pick random values for variables outside of S and look at the remaining polynomial as an univariate one, interpolate it to get its degree

Partial degree

We want to determine the degree of a subset S of variables of the polynomial.

1. pick random values for variables outside of S and look at the remaining polynomial as an univariate one, interpolate it to get its degree
2. evaluate the polynomial on a large value for the variables of S and small random values for the others

Partial degree

We want to determine the degree of a subset S of variables of the polynomial.

1. pick random values for variables outside of S and look at the remaining polynomial as an univariate one, interpolate it to get its degree
2. evaluate the polynomial on a large value for the variables of S and small random values for the others
3. if the polynomial is given by a circuit, transform it into its homogeneous components with regard to S

Partial degree

We want to determine the degree of a subset S of variables of the polynomial.

1. pick random values for variables outside of S and look at the remaining polynomial as an univariate one, interpolate it to get its degree
2. evaluate the polynomial on a large value for the variables of S and small random values for the others
3. if the polynomial is given by a circuit, transform it into its homogeneous components with regard to S

These algorithms are randomized (again the error is exponentially small), and in polynomial time in the number of variables and the degree.

Partial degree

We want to determine the degree of a subset S of variables of the polynomial.

1. pick random values for variables outside of S and look at the remaining polynomial as an univariate one, interpolate it to get its degree
2. evaluate the polynomial on a large value for the variables of S and small random values for the others
3. if the polynomial is given by a circuit, transform it into its homogeneous components with regard to S

These algorithms are randomized (again the error is exponentially small), and in polynomial time in the number of variables and the degree.

Improving the delay

Partial-Monomial
Input: a polynomial given as a black box and two sets of variables L_1 and L_2
Output: accept if there is a monomial in the polynomial in which no variables of L_1 appear, but all of those of L_2 do.

When the polynomial is **multilinear**, this problem can be solved by finding the degree of $P_{\bar{L_1}}$ with regard to L_2: test if the degree is equal to $|L_2|$.

Improving the delay

PARTIAL-MONOMIAL
Input: a polynomial given as a black box and two sets of variables L_1 and L_2
Output: accept if there is a monomial in the polynomial in which no variables of L_1 appear, but all of those of L_2 do.

When the polynomial is **multilinear**, this problem can be solved by finding the degree of $P_{\bar{L_1}}$ with regard to L_2: test if the degree is equal to $|L_2|$.

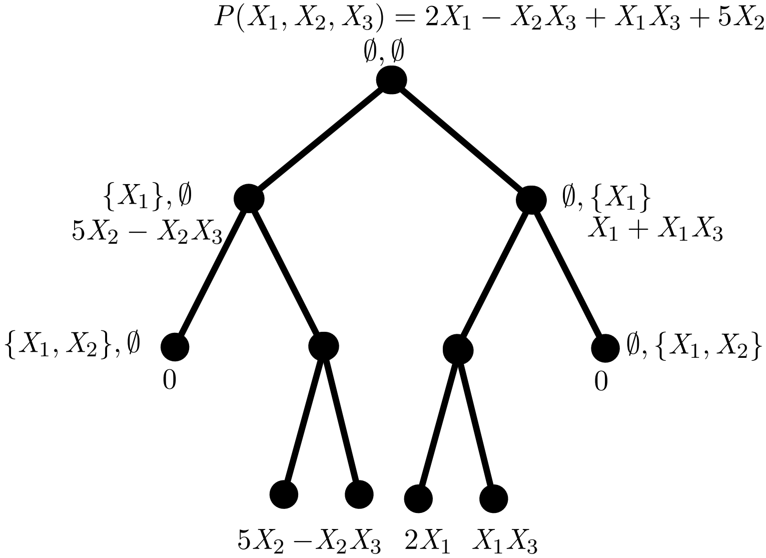
Use this procedure for a depth first traversal of a tree whose leaves are the monomials.

Improving the delay

Partial-Monomial
Input: a polynomial given as a black box and two sets of variables L_1 and L_2
Output: accept if there is a monomial in the polynomial in which no variables of L_1 appear, but all of those of L_2 do.

When the polynomial is **multilinear**, this problem can be solved by finding the degree of $P_{\bar{L_1}}$ with regard to L_2: test if the degree is equal to $|L_2|$.

Use this procedure for a depth first traversal of a tree whose leaves are the monomials.


$P(X_1, X_2, X_3) = 2X_1 - X_2X_3 + X_1X_3 + 5X_2$
$\emptyset, \emptyset$
$\{X_1\}, \emptyset$
$5X_2 - X_2X_3$
$\emptyset, \{X_1\}$
$X_1 + X_1X_3$
$\{X_1, X_2\}, \emptyset$
0
$\emptyset, \{X_1, X_2\}$
0
$5X_2$
$-X_2X_3$
$2X_1$
X_1X_3

Polynomial delay algorithm

Theorem

Let P be a multilinear polynomial with n variables and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1 - \epsilon$ and a delay ***polynomial*** *in n, D and $\log(\epsilon)^{-1}$.*

Polynomial delay algorithm

Theorem

Let P be a multilinear polynomial with n variables and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1 - \epsilon$ and a delay ***polynomial*** *in n, D and $\log(\epsilon)^{-1}$.*

- The algorithm can be parallelized.

Polynomial delay algorithm

Theorem

Let P be a multilinear polynomial with n variables and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1-\epsilon$ and a delay ***polynomial*** *in n, D and $\log(\epsilon)^{-1}$.*

- The algorithm can be parallelized.
- It works on finite fields of small characteristic (can be used to speed up computation).

Polynomial delay algorithm

Theorem

*Let P be a multilinear polynomial with n variables and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1 - \epsilon$ and a delay **polynomial** in n, D and $\log(\epsilon)^{-1}$.*

- The algorithm can be parallelized.
- It works on finite fields of small characteristic (can be used to speed up computation).
- On classes of polynomials given by circuits on which PIT can be derandomized, this algorithm also can be derandomized. STOC 2010, Karnin, Mukhopadhyay, Shpilka, Volkovich: deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in

Polynomial delay algorithm

Theorem

Let P be a multilinear polynomial with n variables and a total degree D. There is an algorithm which computes the set of monomials of P with probability $1 - \epsilon$ and a delay ***polynomial*** *in n, D and $\log(\epsilon)^{-1}$.*

- The algorithm can be parallelized.
- It works on finite fields of small characteristic (can be used to speed up computation).
- On classes of polynomials given by circuits on which PIT can be derandomized, this algorithm also can be derandomized. STOC 2010, Karnin, Mukhopadhyay, Shpilka, Volkovich: deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in

Comparison to other algorithms

	Ben-Or Tiwari	Zippel	KS	My Algorithm
Algorithm type	Deterministic	Probabilistic	Probabilistic	Probabilistic
Number of calls	$2T$	tnD	tn^7D^4	$tnD(n + \log(\epsilon^{-1}))$
Total time	Quadratic in T	Quadratic in t	Quadratic in t	Linear in t
Enumeration	Exponential	**TotalPP**	**IncPP**	**DelayPP**
Size of points	$T \log(n)$	$\log(nT^2\epsilon^{-1})$	$\log(nD\epsilon^{-1})$	$\log(D)$

Figure: Comparison of interpolation algorithms on multilinear polynomials

Good total time and best delay, but only on multilinear polynomials.

Comparison to other algorithms

	Ben-Or Tiwari	Zippel	KS	My Algorithm
Algorithm type	Deterministic	Probabilistic	Probabilistic	Probabilistic
Number of calls	$2T$	tnD	tn^7D^4	$tnD(n + \log(\epsilon^{-1}))$
Total time	Quadratic in T	Quadratic in t	Quadratic in t	Linear in t
Enumeration	Exponential	**TotalPP**	**IncPP**	**DelayPP**
Size of points	$T \log(n)$	$\log(nT^2\epsilon^{-1})$	$\log(nD\epsilon^{-1})$	$\log(D)$

Figure: Comparison of interpolation algorithms on multilinear polynomials

Good total time and best delay, but only on multilinear polynomials.

Higher degree polynomials

Interpolating a polynomial of degree d?

Only known methods use the trick of substracting a monomial: **incremental time**.

Aim: reducing the number of calls to the black-box at each step.

Higher degree polynomials

Interpolating a polynomial of degree d?

Only known methods use the trick of substracting a monomial: **incremental time**.

Aim: reducing the number of calls to the black-box at each step.

- KS algorithm: $O(n^7 D^4)$ calls

Higher degree polynomials

Interpolating a polynomial of degree d?

Only known methods use the trick of substracting a monomial: **incremental time**.

Aim: reducing the number of calls to the black-box at each step.

- KS algorithm: $O(n^7 D^4)$ calls
- The idea of the two presented algorithms combined to find the "highest" degree polynomial: $O(n^2 D^{d-1})$ calls

Higher degree polynomials

Interpolating a polynomial of degree d?

Only known methods use the trick of substracting a monomial: **incremental time**.

Aim: reducing the number of calls to the black-box at each step.

- KS algorithm: $O(n^7 D^4)$ calls
- The idea of the two presented algorithms combined to find the "highest“ degree polynomial: $O(n^2 D^{d-1})$ calls

Open question: how to efficiently represent and compute the partial polynomial at each step? Easier with circuits, formula, polynomial of low degree, over fixed finite fields ?

Higher degree polynomials

Interpolating a polynomial of degree d?

Only known methods use the trick of substracting a monomial: **incremental time**.

Aim: reducing the number of calls to the black-box at each step.

- KS algorithm: $O(n^7 D^4)$ calls
- The idea of the two presented algorithms combined to find the "highest" degree polynomial: $O(n^2 D^{d-1})$ calls

Open question: how to efficiently represent and compute the partial polynomial at each step? Easier with circuits, formula, polynomial of low degree, over fixed finite fields ?

Limits to efficient interpolation

NON-ZERO-MONOMIAL
Input: a polynomial and a term $\vec{X}^{\vec{e}}$
Output: accept if $\vec{X}^{\vec{e}}$ has a coefficient different from zero in the polynomial

MONOMIAL-COEFFICIENT
Input: a polynomial given as a circuit and a term $\vec{X}^{\vec{e}}$
Output: return the coefficient of $\vec{X}^{\vec{e}}$ in the polynomial

Limits to efficient interpolation

NON-ZERO-MONOMIAL
Input: a polynomial and a term $\vec{X}^{\vec{e}}$
Output: accept if $\vec{X}^{\vec{e}}$ has a coefficient different from zero in the polynomial

MONOMIAL-COEFFICIENT
Input: a polynomial given as a circuit and a term $\vec{X}^{\vec{e}}$
Output: return the coefficient of $\vec{X}^{\vec{e}}$ in the polynomial

Algorithm similar to the polynomial delay one can solve both these problems.

Limits to efficient interpolation

NON-ZERO-MONOMIAL
Input: a polynomial and a term $\vec{X}^{\vec{e}}$
Output: accept if $\vec{X}^{\vec{e}}$ has a coefficient different from zero in the polynomial

MONOMIAL-COEFFICIENT
Input: a polynomial given as a circuit and a term $\vec{X}^{\vec{e}}$
Output: return the coefficient of $\vec{X}^{\vec{e}}$ in the polynomial

Algorithm similar to the polynomial delay one can solve both these problems.

Idea: if they are hard for some family of easy to compute polynomials, the polynomial delay interpolation should also be hard

Limits to efficient interpolation

NON-ZERO-MONOMIAL
Input: a polynomial and a term $\vec{X}^{\vec{e}}$
Output: accept if $\vec{X}^{\vec{e}}$ has a coefficient different from zero in the polynomial

MONOMIAL-COEFFICIENT
Input: a polynomial given as a circuit and a term $\vec{X}^{\vec{e}}$
Output: return the coefficient of $\vec{X}^{\vec{e}}$ in the polynomial

Algorithm similar to the polynomial delay one can solve both these problems.

Idea: if they are hard for some family of easy to compute polynomials, the polynomial delay interpolation should also be hard

Degree n polynomial

Proposition

The problem MONOMIAL-COEFFICIENT *is* $\#\mathrm{P}$*-hard.*

Proof.

$$Q(X, Y) = \prod_{i=1}^{n} (\sum_{j=1}^{n} X_{i,j} Y_j)$$

The term $T = \prod_{j=1}^{n} Y_j$ has $\sum_{\sigma \in \Sigma_n} \prod_{i=1}^{n} X_{i,\sigma(i)}$ for coefficient, which is the Permanent in the variables $X_{i,j}$. □

Degree n polynomial

Proposition

The problem MONOMIAL-COEFFICIENT *is* $\#\mathrm{P}$*-hard.*

Proof.

$$Q(X, Y) = \prod_{i=1}^{n} (\sum_{j=1}^{n} X_{i,j} Y_j)$$

The term $T = \prod_{j=1}^{n} Y_j$ has $\sum_{\sigma \in \Sigma_n} \prod_{i=1}^{n} X_{i,\sigma(i)}$ for coefficient, which is the Permanent in the variables $X_{i,j}$. □

Degree 3 polynomial

Proposition

The problem NON-ZERO-MONOMIAL *restricted to degree* 3 *polynomials is* NP*-hard.*

Proof.

Reduction from EXACT-COVER:

$$\prod_{\{i,j,k\} \in C} (X_i X_j X_k + 1)$$

There is an exact cover if $\prod_{i \in [n]} X_i$ has a coefficient different from zero $\square$

Degree 3 polynomial

Proposition

The problem Non-Zero-Monomial *restricted to degree* 3 *polynomials is* NP*-hard.*

Proof.

Reduction from Exact-Cover:

$$\prod_{\{i,j,k\}\in C} (X_i X_j X_k + 1)$$

There is an exact cover if $\prod_{i\in[n]} X_i$ has a coefficient different from zero □

Degree 2 polynomial

Proposition

The problem Non-Zero-Monomial *restricted to degree* 2 *polynomials is* NP-*hard.*

Proof.

Reduction from Hamiltonian Path over degree 2 directed graphs. Use a polynomial derived from the Matrix Tree theorem. Use Non-Zero-Monomial on a polynomial number of terms of this polynomial, if one is in there is a spanning tree which is also an Hamiltonian path. □

Degree 2 polynomial

Proposition

The problem NON-ZERO-MONOMIAL *restricted to degree* 2 *polynomials is* NP*-hard.*

Proof.

Reduction from HAMILTONIAN PATH over degree 2 directed graphs. Use a polynomial derived from the Matrix Tree theorem. Use NON-ZERO-MONOMIAL on a polynomial number of terms of this polynomial, if one is in there is a spanning tree which is also an Hamiltonian path. □

- These proofs are for polynomials of small degree and (except the last) given by small depth circuits!

Degree 2 polynomial

Proposition

The problem Non-Zero-Monomial *restricted to degree* 2 *polynomials is* NP*-hard.*

Proof.

Reduction from Hamiltonian Path over degree 2 directed graphs. Use a polynomial derived from the Matrix Tree theorem. Use Non-Zero-Monomial on a polynomial number of terms of this polynomial, if one is in there is a spanning tree which is also an Hamiltonian path. □

- These proofs are for polynomials of small degree and (except the last) given by small depth circuits!
- Conclusion: some monomials are harder than others.

Degree 2 polynomial

Proposition

The problem NON-ZERO-MONOMIAL *restricted to degree* 2 *polynomials is* NP*-hard.*

Proof.

Reduction from HAMILTONIAN PATH over degree 2 directed graphs. Use a polynomial derived from the Matrix Tree theorem. Use NON-ZERO-MONOMIAL on a polynomial number of terms of this polynomial, if one is in there is a spanning tree which is also an Hamiltonian path. □

- These proofs are for polynomials of small degree and (except the last) given by small depth circuits!
- Conclusion: some monomials are harder than others.
- Question of Kayal: what is the complexity of computing the leading monomial of a depth three circuit?

Degree 2 polynomial

Proposition

The problem NON-ZERO-MONOMIAL *restricted to degree* 2 *polynomials is* NP*-hard.*

Proof.

Reduction from HAMILTONIAN PATH over degree 2 directed graphs. Use a polynomial derived from the Matrix Tree theorem. Use NON-ZERO-MONOMIAL on a polynomial number of terms of this polynomial, if one is in there is a spanning tree which is also an Hamiltonian path. □

- These proofs are for polynomials of small degree and (except the last) given by small depth circuits!
- Conclusion: some monomials are harder than others.
- Question of Kayal: what is the complexity of computing the leading monomial of a depth three circuit?

Thank for listening!

Shameless self-promotion

I am a new Post-doc here, working with Pascal Koiran and Natacha Portier.
Interested to work in complexity in general and especially:

- decomposition of matroids, hypergraphs and other structures (width notions)
- circuit complexity
- enumeration complexity
- implicit complexity