Enumeration of the monomials of a polynomial and related complexity classes

Yann Strozecki

Équipe de Logique Mathématique, Paris 7

We are interested by **enumeration problems**.

We are interested by **enumeration problems**.

$A(x, y)$ is a predicate.

We are interested by **enumeration problems**.

$A(x, y)$ is a predicate.

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

We are interested by **enumeration problems**.

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

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

Example

$A(x, y)$ means y is a perfect matching in the graph x. The decision problem is to decide if there is a perfect matching.

We are interested by **enumeration problems**.

$A(x, y)$ is a predicate.

$\sharp\{y|A(x,y)\}$: counting problem

We are interested by **enumeration problems**.

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

$\sharp\{y|A(x, y)\}$: counting problem

Example

$A(x, y)$ means y is a perfect matching in the graph x. The counting problem is to count the number of perfect matchings.

We are interested by **enumeration problems**.

$A(x, y)$ is a predicate.

$\{y|A(x, y)\}$: enumeration problem

We are interested by **enumeration problems**.

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

$\{y|A(x, y)\}$: enumeration problem

Example

$A(x, y)$ means y is a perfect matching in the graph x. The enumeration problem is to find every perfect matching.

For enumeration problems we have two interesting complexity measures:

For enumeration problems we have two interesting complexity measures:

1. the total time

For enumeration problems we have two interesting complexity measures:

1. the total time related to the the number of solutions

For enumeration problems we have two interesting complexity measures:

1. the total time related to the the number of solutions
2. the delay

The problem :

The problem :

A black box which computes $P(X_1, \ldots, X_n)$.

The problem :

A black box which computes $P(X_1, \ldots, X_n)$.

What is P ?

The problem : Is it an enumeration problem ?

A black box which computes $P(X_1, \ldots, X_n)$.

What is P ?

The problem : Is it an enumeration problem ?

A black box which computes $P(X_1, \ldots, X_n)$.

What is P ? The set of monomials of P

The problem is known as the **interpolation problem**.

The problem is known as the **interpolation problem**.

- Deterministic method : it relies on evaluation on very large primes

The problem is known as the **interpolation problem**.

- Deterministic method : it relies on evaluation on very large primes
- Probabilistic method : it relies on the Schwarz-Zippel lemma and the solving of large linear system

The problem is known as the **interpolation problem**.

- Deterministic method : it relies on evaluation on very large primes
- Probabilistic method : it relies on the Schwarz-Zippel lemma and the solving of large linear system

The complexity of all algorithms depends on the number of monomials and often need an a priori bound on this number.

Lemma (Schwarz-Zippel)

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

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if we chose randomly $x_1, \ldots, x_n$ 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.

Probabilistic algorithm for the *Zero Avoidance Problem*.

Lemma (Schwarz-Zippel)

Let P be a non zero polynomial with n variables of total degree D, if we chose randomly $x_1, \ldots, x_n$ 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.

Probabilistic algorithm for the *Zero Avoidance Problem*.

Two ways of improving the probability : big evaluation points or repetition

$L \subseteq [|1, n|]$ is a set of indices of variables.
We note P_L the polynomial P with all variables with indices outside of L set to 0.

$L \subseteq [|1, n|]$ is a set of indices of variables.
We note P_L the polynomial P with all variables with indices outside of L set to 0.

Lemma

Let P be a multilinear polynomial without constant term and L a minimal set of variables such that P_L is not identically zero. Then there is an integer λ such that $P_L = \lambda \vec{X}^L$.

$L \subseteq [|1, n|]$ is a set of indices of variables.
We note P_L the polynomial P with all variables with indices outside of L set to 0.

Lemma

Let P be a multilinear polynomial without constant term and L a minimal set of variables such that P_L is not identically zero. Then there is an integer λ such that $P_L = \lambda \vec{X}^L$.

From now on we assume that the polynomials are **multilinear** without constant term.

We build a set of variable L :

Input : A n variables black box polynomial P

For $i = 1$ **to** n **do**
If not_zero($P_{L \setminus \{i\}}$)
Then $L = L \setminus \{i\}$

We build a set of variable L :

Input : A n variables black box polynomial P

For $i = 1$ **to** n **do**
If not_zero($P_{L\setminus\{i\}}$)
Then $L = L \setminus \{i\}$

After this loop, P_L is non zero and L is minimal, with high probability.

$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3 \text{ and } L = \{1, 2, 3\}$$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3 \text{ and } L = \{1, 2, 3\}$$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$L = \{3\}$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

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

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

- $L = \{2, 3\}$ evaluation of P_L on $X_2 = 1, X_3 = -1 \rightarrow 0$
- $L = \{1, 3\}$ evaluation of P_L on $X_1 = 1, X_3 = 1$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

- $L = \{2, 3\}$ evaluation of P_L on $X_2 = 1, X_3 = -1 \rightarrow 0$
- $L = \{1, 3\}$ evaluation of P_L on $X_1 = 1, X_3 = 1 \quad \rightarrow 1$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

- $L = \{2, 3\}$ evaluation of P_L on $X_2 = 1, X_3 = -1 \rightarrow 0$
- $L = \{1, 3\}$ evaluation of P_L on $X_1 = 1, X_3 = 1 \rightarrow 1$
- $L = \{1\}$ evaluation of P_L on $X_1 = 1$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

- $L = \{2, 3\}$ evaluation of P_L on $X_2 = 1, X_3 = -1 \rightarrow 0$
- $L = \{1, 3\}$ evaluation of P_L on $X_1 = 1, X_3 = 1 \rightarrow 1$
- $L = \{1\}$ evaluation of P_L on $X_1 = 1 \rightarrow 0$

$P(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2 + X_3$ and $L = \{1, 2, 3\}$

- $L = \{2, 3\}$ evaluation of P_L on $X_2 = 1, X_3 = -1 \rightarrow 0$
- $L = \{1, 3\}$ evaluation of P_L on $X_1 = 1, X_3 = 1 \rightarrow 1$
- $L = \{1\}$ evaluation of P_L on $X_1 = 1 \rightarrow 0$

$L = \{1, 3\}$

Theorem

The algorithm finds a monomial of a multilinear polynomial given as a black box, with probability $1 - \epsilon$*, by making* $O(n \log(\frac{n}{\epsilon}))$ *calls to the black box on entries of size* $\log(2D)$.

Theorem

The algorithm finds a monomial of a multilinear polynomial given as a black box, with probability $1 - \epsilon$, by making $O(n \log(\frac{n}{\epsilon}))$ calls to the black box on entries of size $\log(2D)$.

Errors only appear in the procedure not_zero with probability $\frac{\epsilon}{n+1}$.

Theorem

The algorithm finds a monomial of a multilinear polynomial given as a black box, with probability $1-\epsilon$, by making $O(n\log(\frac{n}{\epsilon}))$ calls to the black box on entries of size $\log(2D)$.

Errors only appear in the procedure not_zero with probability $\frac{\epsilon}{n+1}$.

We use this procedure $n+1$ times : we can bound the total probability of error by ϵ.

We simulate the polynomial $P - Q$ when P is given by a black box and Q explicitely by *subtract*(P, Q).

We simulate the polynomial $P - Q$ when P is given by a black box and Q explicitely by *subtract*(P, Q).

Input : A n variables black box polynomial P

$Q \longleftarrow 0$
While *not_zero(subtract*(P,Q)*)*
$M \longleftarrow$ *find_monomial(subtract*(P, Q)*)*
Write(M)
$Q \longleftarrow Q + M$

Theorem

Let P be a multilinear polynomial with n variables, t monomials, C a bound on the size of its coefficient and D its total degree. Previous algorithm computes the set of monomials of P with probability $1-\epsilon$. It does $O(tn(n+\log(\frac{1}{\epsilon})))$ calls to the oracle on points of size $2D$. The delay between the i^{th} and $i+1^{th}$ found monomials is bounded by $O(iD\max(C,D)n(n+\log(\frac{1}{\epsilon})))$.

L_1 and L_2 are two disjoint sets of indices of variables.
Does P contains a monomial $X^{\vec{e}}$ whose support has no intersection with L_1 but contains L_2 ?

L_1 and L_2 are two disjoint sets of indices of variables.
Does P contains a monomial $X^{\vec{e}}$ whose support has no intersection with L_1 but contains L_2 ?

We have the equality $P_{\bar{L_1}} = \vec{X}^{L_2} P_1(\vec{X}) + P_2(\vec{X})$, by Euclidean division.
Previous question is equivalent to is P_1 zero ?

We assume that the polynomial is multilinear and its coefficents are positive and of size bounded by C.

We assume that the polynomial is multilinear and its coefficents are positive and of size bounded by C.

A good choice of evaluation points :

$$\begin{cases} x_i = 0 & \text{if } i \in L_1 \\ x_i = 2^{n+C} & \text{if } i \in L_2 \\ x_i = 1 & \text{else} \end{cases}$$

We assume that the polynomial is multilinear and its coefficents are positive and of size bounded by C.

A good choice of evaluation points :

$$\begin{cases} x_i = 0 & \text{if } i \in L_1 \\ x_i = 2^{n+C} & \text{if } i \in L_2 \\ x_i = 1 & \text{else} \end{cases}$$

$P = (2^{n+C})^l P_1(\vec{x}) + P_2(\vec{x})$

If P_1 is zero, $P(\vec{x}) < 2^{l(n+C)}$

If P_1 is not zero, $P(\vec{x}) \geq 2^{l(n+C)}$

If P_1 is zero, $P(\vec{x}) < 2^{l(n+C)}$

If P_1 is not zero, $P(\vec{x}) \geq 2^{l(n+C)}$

We can decide the question does P contains a monomial $X^{\vec{e}}$ whose support has no intersection with L_1 but contains L_2, with one call to the oracle.
We call this procedure *not_zero_improved*(L_1, L_2, P).

A depth first search to enumerate all monomials :

$Monomial(L_1, L_2, i) =$
If $i = n + 1$
Write The monomial of support L_2
If $not\_zero\_improved(L_1 \cup \{i\}, L_2, P)$
Then $Monomial(L_1 \cup \{i\}, L_2, i + 1)$
If $not\_zero\_improved(L_1, L_2 \cup \{i\}, P)$
Then $Monomial(L_1, L_2 \cup \{i\}, i + 1$
$in\ Monomial(\varnothing, \varnothing, 0)$

Theorem

Let P be a multilinear polynomial with n variables and positive coefficents of size C, t monomials and D its total degree. Previous algorithm computes the set of monomials of P. It does $O(tn)$ calls to the oracle on points of size $O(C+n)$. The delay between the i^{th} and $i+1^{th}$ found monomials is bounded by a time $O(n(C+n))$ and $O(n)$ oracle calls.

Theorem

Let P be a multilinear polynomial with n variables and positive coefficents of size C, t monomials and D its total degree. Previous algorithm computes the set of monomials of P. It does $O(tn)$ calls to the oracle on points of size $O(C+n)$. The delay between the i^{th} and $i+1^{th}$ found monomials is bounded by a time $O(n(C+n))$ and $O(n)$ oracle calls.

The algorithm is easily generalizable to polynomials with arbitrary coefficients, if we make it probabilistic.

First algorithm :

- evaluation points of size $log(D)$
- incremental delay
- we can relax some hypothesis

First algorithm :

- evaluation points of size $log(D)$
- incremental delay
- we can relax some hypothesis

No two monomials of the polynomial have the same support.
It is verified when the polynomial is multilinear.

First algorithm :

- evaluation points of size $log(D)$
- incremental delay
- we can relax some hypothesis

Second algorithm :

- evaluation points of size polynomial in n
- poynomial delay
- easy to paralellize

Example

Let G be a graph with n vertices, we define an $n \times n$ matrix M such that $M_{i,j} = x_{i,j}$ if and only if (i,j) is an edge in G. We associate to G the multilinear polynomial $\det(M)$, whose monomials represents cycle covers of G. The problem of enumerating the monomials is equivalent to enumerating the cycle covers of a graph, which seems a natural problem.

Definition

An enumeration problem A is decidable in probabilistic polynomial total time, written **TotalPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability greater than $\frac{2}{3}$ and satisfies for all x, $T(x, |M(x)|) < Q(|x|, |M(x)|)$.

Definition

An enumeration problem A is decidable in probabilistic polynomial total time, written **TotalPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability greater than $\frac{2}{3}$ and satisfies for all x, $T(x, |M(x)|) < Q(|x|, |M(x)|)$.

Algorithm of the litterature applied to the example = **TotalPP**.

Definition

An enumeration problem A is decidable in probabilistic incremental polynomial time, written **IncPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability $\frac{2}{3}$ and satisfies for all x, $T(x, i+1) - T(x, i) \leq Q(|x|, i)$.

Definition

An enumeration problem A is decidable in probabilistic incremental polynomial time, written **IncPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability $\frac{2}{3}$ and satisfies for all x, $T(x, i+1) - T(x, i) \leq Q(|x|, i)$.

Proposition

$\textsc{AnotherSolution}_A$ *has a solution in probabilistic polynomial time if and only if* $A \in$ **IncPP**.

Definition

An enumeration problem A is decidable in probabilistic incremental polynomial time, written **IncPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability $\frac{2}{3}$ and satisfies for all x, $T(x, i+1) - T(x, i) \leq Q(|x|, i)$.

Proposition

$\textsc{AnotherSolution}_A$ *has a solution in probabilistic polynomial time if and only if* $A \in$ **IncPP**.

First algorithm applied to the example = **IncPP**.

Definition

An enumeration problem A is decidable in probabilistic polynomial delay, written **DelayPP**, if there is a polynomial $Q(x,y)$ and a machine M which solves A with probability $\frac{2}{3}$ and satisfies for all x and all i, $T(x,i+1) - T(x,i) \leq Q(|x|)$.

Definition

An enumeration problem A is decidable in probabilistic polynomial delay, written **DelayPP**, if there is a polynomial $Q(x, y)$ and a machine M which solves A with probability $\frac{2}{3}$ and satisfies for all x and all i, $T(x, i+1) - T(x, i) \leq Q(|x|)$.

Second algorithm applied to the example $=$ **DelayPP**.

Notion of spanning hyertree in hypergraph.

Notion of spanning hyertree in hypergraph.

A polynomial Z defined for each 3-uniform hypergraph with coefficients -1 or 1, whose monomials are in bijection with the spanning hypertrees of the hypergraph.

Notion of spanning hyertree in hypergraph.

A polynomial Z defined for each 3-uniform hypergraph with coefficients -1 or 1, whose monomials are in bijection with the spanning hypertrees of the hypergraph.

It has been proved that Z is the Pfaffian of a matrix, whose coefficients are linear polynomials depending on the hypergraph.

Notion of spanning hyertree in hypergraph.

A polynomial Z defined for each 3-uniform hypergraph with coefficients -1 or 1, whose monomials are in bijection with the spanning hypertrees of the hypergraph.

It has been proved that Z is the Pfaffian of a matrix, whose coefficients are linear polynomials depending on the hypergraph.

The enumeration of the spanning hypertrees of a 3-uniform hypergraph is in **DelayPP**.

By combining the two algorithms we can find the monomials of a degree 2 polynomials.

By combining the two algorithms we can find the monomials of a degree 2 polynomials.

Question : is it possible to have an incremental algorithm for degree 3 or more ?

$S = [|1, n|]$ is a set of size n and C be a collection of three elements subsets of S. $C' \subseteq C$, $\chi(C') = \prod_{\{i,j,k\} \in C'} X_i X_j X_k$.

$S = [|1, n|]$ is a set of size n and C be a collection of three elements subsets of S. $C' \subseteq C$, $\chi(C') = \prod_{\{i,j,k\} \in C'} X_i X_j X_k$.

P_C is the sum of the $\chi(C')$ for all subsets C'. The degree of P_C is the maximal number of occurences of an element in C.

$S = [|1, n|]$ is a set of size n and C be a collection of three elements subsets of S. $C' \subseteq C$, $\chi(C') = \prod_{\{i,j,k\} \in C'} X_i X_j X_k$.

P_C is the sum of the $\chi(C')$ for all subsets C'. The degree of P_C is the maximal number of occurences of an element in C.

$P_C = \prod_{\{i,j,k\} \in C} (X_i X_j X_k + 1)$, which makes it easy to evaluate in polynomial time.

$S = [|1, n|]$ is a set of size n and C be a collection of three elements subsets of S. $C' \subseteq C$, $\chi(C') = \prod_{\{i,j,k\} \in C'} X_i X_j X_k$.

P_C is the sum of the $\chi(C')$ for all subsets C'. The degree of P_C is the maximal number of occurences of an element in C.

$P_C = \prod_{\{i,j,k\} \in C} (X_i X_j X_k + 1)$, which makes it easy to evaluate in polynomial time.

Remark

A subset C' is an exact cover of S if and only if $\chi(C') = \prod_{i \in S} X_i$.

Assume we have a generalization of the polynomial delay algorithm for degree 3 polynomials : it allows us to test if there is a precise monomial in a polynomial in probabilistic polynomial time.

Assume we have a generalization of the polynomial delay algorithm for degree 3 polynomials : it allows us to test if there is a precise monomial in a polynomial in probabilistic polynomial time.

Then we can decide if $\prod_{i \in S} X_i$ is in P_C, which is of degree 3 if no elements of S occurs in more than three elements of C. The problem of finding an exact cover even if no element occurs in more than three subsets is NP-complete : it implies that $\mathrm{RP} = \mathrm{NP}$.

Conjecture : no polynomial delay algorithm for degree 2 or more

Conjecture : no polynomial delay algorithm for degree 2 or more

Conjecture : no incremental algorithm for degree 3 or more

Thanks for listening!