30. Algebraization

Goal: Transform problems into an algebraic format,
then solve the corresponding algebraic problems.

Background:
- HAMILTONIAN has been shown to be solvable (using algebraization)
  in time $O^*(1.657^n)$ by Andreas Björklund in 2010.
- The bound had not been improved over $O^*(2^n)$ since 1962.
- The tools are only available since 2009,
  have been developed on k-PATH.

Approach:
- Formulate combinatorial problems
  as the existence of certain monomials in a polynomial.
  This is called algebraization.
- Algebraic detection (of monomials) means to
  extract information about a polynomial
  by assigning values to the variables.
- Assignments that fit our purposes lead to
  calculations modulo 2.
- Calculations modulo lead to unwanted cancellations.
  Solve this problem with fingerprints
  that augment the polynomial.

[Not here: Exploit cancellations
Treat computation modulo as a resource to that
many unwanted monomials cancel out.]

Sample problem on which we illustrate algebraization:

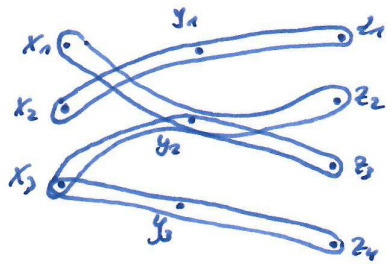
k-3D-MATCHING: captain of a plane / 2nd officer / destination.

Given: Triples $T \subseteq X \times Y \times Z$

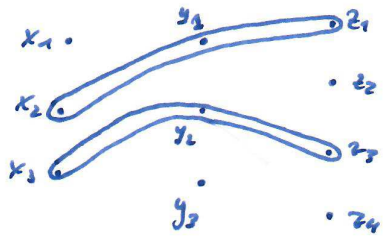
Parameter: $k \in \mathbb{N}$

Question: $\exists T' \subseteq T: |T'| \geq k$ and triples in T' disjoint
(in all components).

Example:



Solution for $k=2$:



30.1 Algebraization

30.1.1 A Polynomial for 3D-Matching

- View each element of the sets X, Y, Z as a variable.

  So $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2, z_3, z_4\}$.

- For each triple, construct a monomial,
  a product of variables.
  A monomial is called multi-linear
  if every variable occurs at most once.
  In the example:

  $\{x_1 y_2 z_2,\ x_2 y_1 z_1,\ x_3 y_2 z_3,\ x_3 y_3 z_4\}$.

- The instance polynomial P_1 is the sum of the monomials:

  $$P_1 = x_1 y_2 z_2 + x_2 y_1 z_1 + x_3 y_2 z_3 + x_3 y_3 z_4.$$

- Set $P_k = P_1^k$, called the kth encoding-polynomial.
  We assume P_1^k is computed in a commutative ring $(R, +, \cdot)$,
  so $\cdot$ $(R, +)$ is a commutative group,
  $\cdot$ $(R, \cdot)$ is a commutative monoid, and
  $\cdot$ multiplication distributes over addition.

  In the example:

  $$\begin{aligned} P_2 = P_1^2 = {} & (x_1 y_2 z_2)^2 + (x_2 y_1 z_1)^2 + (x_3 y_2 z_3)^2 + (x_3 y_3 z_4)^2 \\ & + 2 x_1 x_2 y_1 y_2 z_1 z_2 + 2 x_1 x_3 y_2^2 z_2 z_3 \\ & + 2 x_1 x_3 y_2 y_3 z_2 z_4 + 2 x_2 x_3 y_1 y_2 z_1 z_3 \\ & + 2 x_2 x_3 y_1 y_3 z_1 z_4 + 2 x_3^2 y_2 y_3 z_3 z_4. \end{aligned}$$

- The latter is called the sum-product expansion.

- In the so-called solution part of the sum-product expansion, monomials are multi-linear.
  In the non-solution part, monomials contain squares or other higher powers of variables.

Lemma:
The instance of k-3D-MATCHING contains a solution of size $\geq k$
iff
the sum-product expansion of the kth encoding polynomial P_k contains a multi-linear monomial.

Note:
- It is sufficient to compute the sum-product expansion of P_k.
- Ref: The $O^*(2^N)$ with $N = |X| + |Y| + |Z|$ algorithm is known for containment of a multi-linear monomial.
- The number of monomials of length $3k$ may be much larger (we may have $N < 3k$).

30.1.2 The Dynamic Programming Algebra

Idea:
- Expand P_k into sum-product form slowly, one multiplication at a time.
- Monomials containing squared variables do not affect the presence of multi-linear monomials.
  $\Rightarrow$ Throw them away!
- There are 2^N multi-linear monomials in N variables.
  The truncated expansion is thus computable in $O^*(2^N)$.

Formally:
- Compute the sum-product expansion in a dynamic programming algebra of polynomials.
- Use the rules for commutative rings plus squared variables evaluate to 0.
- Note that now the non-multi-linear monomials evaluate to 0.

Corollary:

An instance of k-3D-MATCHING contains a solution of size $\geq k$

iff

$P_k \neq 0$ in the dynamic programming algebra.

Note that also $N \geq 3k$ is possible.
However, we are only looking for a matching with k triples.
So

targetting $O^*(2^{3k})$ makes sense.

Testing Polynomials: Testing whether an arithmetic expression equals the 0-polynomial will be a recurring theme in this lecture.

30.1.3 Parameterizing via Assignment

Goal: Demonstrate the use of assignments to the variables to extract information from the kth encoding polynomial.

Problem: Number of variables N may be large.

Approach: Color coding ?

- In the example, we need multi-linear monomials which have six distinct variables.
- Introduce a new set

  $W = \{w_1, \ldots, w_6\}$

  of six variables.
- Pick a random assignment of the variables $X_U Y_U Z$ to W.
- Note that this is like coloring the N variables with 3k colors.

In the example:

$x_1 = w_1$ $\quad y_1 = w_3$ $\quad z_1 = w_6$

$x_2 = w_4$ $\quad y_2 = w_2$ $\quad z_2 = w_6$

$x_3 = w_1$ $\quad y_3 = w_5$ $\quad z_3 = w_1$

$z_4 = w_2$.

Now evaluating $P_2(X,Y,Z)$ with this assignment yields

$$P_2(W) = (w_1 w_2 w_6)^2 + (w_4 w_3 w_6)^2 + (w_1 w_2 w_1)^2 + (w_1 w_5 w_2)^2 + \dots$$

Clearly, monomials are mapped to monomials.

Lemma:

- If a monomial is non-(multi)-linear in $P_k(X,Y,Z)$, it will be non-multi-linear in $P_k(W)$.
- But: Multi-linear monomials in $P_k(X,Y,Z)$ may be mapped to non-multi-linear ones.

- With the lemma, detecting a multi-linear monomial in $P_k(W)$ allows us to infer its presence in $P_k(X,Y,Z)$.
- We can evaluate $P_k(W)$ in the dynamic programming algebra. Because W contains $3k$ variables, this takes $O^*(2^{3k})$-time.
- The success probability, however, is only $\sim \frac{1}{e^{3k}}$.

Hence, we need around $O(e^{3k})$ attempts to arrive at a constant probability independent of k.

Theorem:

The overall running time of this approach is $O^*((2e)^{3k})$.

This is still a factor of e^{3k} away from the target!?

30.2 Algebraic Detection

Goal: Present a truly algebraic method
for detecting multi-linear monomials.
The previous ones were combinatorial,
presented in algebraic guise.

30.2.1 Matrix Assignment

Recall: We gave an assignment to the variables in the encoding polynomial
and then evaluated the result in the dynamic programming algebra.

Goal: Implement the dynamic programming algebra by means of matrices.
Keep the idea of a random assignment.

Requirements
on a set of random matrices to be used
for the detection of multi-linear monomials.

1.) Commutativity of multiplication (we used a commutative ring).

2.) $m^2 = 0$ for all matrices m.

3.) Multi-linear monomials must survive (evaluate to $\neq 0$)
with constant probability, say $1/4$.
It would be good if the resulting matrix had a diagonal $= 1$.

4.) The matrices must be small for computations to be efficient.
Ideally, matrix operations should take time $O^*(2^k)$ or less.

Theorem (The Mod-2 Matrix Fact):

One can construct such matrices,
provided zero means zero modulo 2.

We only need the algorithmic consequences
of the ability to construct this set of matrices,
not the matrices themselves nor their construction.

↳ Evaluating the polynomial : Still takes $O^*(2^k)$.
(for detecting a multi-linear monomial)

↳ Exponentially small probability $\frac{1}{e^k}$ for survival was raised to a constant probability $\frac{1}{4}$.

The latter fact appears to suggest we got rid of the e^k factor and have reached our target of detecting multi-linear monomials in 2^k (and thus solved k-3D-MATCHING in 2^{3k}).

Problem: · The relaxed notion of zero leads to unwanted cancellations.
· $P(M)$ (with M an assignment of variables to matrices) may evaluate to zero although multi-linear monomials may be present.
· Why? Some monomials occur several times, say twice, and thus $2 x_1 x_2 y_1 y_2 z_1 z_2 = 0 \bmod 2$.

3.2.2 Fingerprints

Goal: Tackle the problem of multiple copies of the same monomial.
Idea: Make sure every monomial is unique.
Technically: Use a set of fingerprint variables $A = \{a_1, a_2, \ldots\}$.

In the example:

$$P_2 = (a_1 x_1 y_2 z_2 + a_2 x_2 y_1 z_1 + a_3 x_3 y_2 z_3 + a_4 x_3 y_3 z_4) * (a_5 x_1 y_2 z_2 + a_6 x_2 y_1 z_1 + a_7 x_3 y_2 z_3 + a_8 x_3 y_3 z_4).$$

· Multiply each monomial occurring in a polynomial by a fresh variable.
· Introduction of auxiliary variables leads to (multi-linear) monomials of higher degree.
· Needs larger matrices in the evaluation phase, results in slower algorithm.

Idea: · Use a different assignment for the A-variables, namely to {0,1}.
· Wish to create an odd number of copies of the multi-linear monomials so that they are not trivially zero modulo 2.

Lemma:
By randomly assigning matrices to the variables X, Y, Z and {0,1} to the variables in A
↳ the polynomial will evaluate to ≠0 with constant probability, if the sum-product expansion contains a multi-linear monomial,
↳ and always to 0, otherwise.

Theorem:
We have designed an $O^*(2^{3k})$-time probabilistic algorithm with constant probability for k-3D-MATCHING.

3.2.3 An Algebraic Framework

Observation: We designed an algorithm for a problem more general than 3D-Matching.

k-LIN-MON:
Given: A polynomial P(X) in some concise, non-expanded representation (typically a circuit).
Parameter: $k \in \mathbb{N}$.
Question: Is there a multi-linear monomial of degree k (k variables) in the sum-product expansion?

Applications: Several ones (path and packing problems, see Koutis, ICALP'08.

What is needed for a generalization: · All steps above, including placement of fingerprint variables, generalise to arbitrary polynomials.

- The trick of randomly assigning $\{0,1\}$ does not work for arbitrary polynomials (we were sure to have $P_n = P_n^2$).

- Generalize the approach to other assignments that
  - work for arbitrary polynomials
  - while still being easy to handle computationally.

Solution: Evaluate $P(X, A)$ in two steps:

↳ Let M be the matrices from the Mod-2 Matrix Fact.
Assign them to the variables in X.
This yields $P(M, A)$, where the A variables have not yet been evaluated.

↳ This $P(M, A)$ is a matrix
whose entries are polynomials over A.
We focus on the diagonal (the entries are all the same by Requirement 3).
Let the resulting polynomial be $Q(A)$.

Lemma (By the Mod-2 Matrix Fact):

- If $P(X)$ does not contain a multi-linear monomial, then $Q(A) = 0 \mod 2$.
- If $P(X)$ does contain a multi-linear monomial, then $Q(A) \neq 0 \mod 2$ with probability $\geq 1/4$.

Hence, the remaining problem is identity testing.

IDENTITY:
Given: Polynomial $Q(A)$
Question: $Q(A) = 0 \mod 2$?

Approach: Evaluate $Q(T)$ on some computed assignment to its variables.

Solution: Schwartz-Zippel-Lemma

↳ Intuitively we picked $\{0,1\}$, because with them we can do addition and multiplication that respect modulo 2 arithmetic.

↳ There are other fields that
- allow the same two operations and
- are larger (contain more values).

↳ Pick a field F of size $O(k)$ for the assignment to the fingerprint variables.
Then the number of assignments outnumbers the number of roots of $Q(T)$.

↳ Hence, a random assignment to this field F will result in

$$Q(F) \neq 0$$

and thus $Q(T) \neq 0 \mod 2$ with high probability.

Theorem:
k-LIN-MON can be solved in $O^*(2^k)$ with constant probability.

Remark:
- Most problems that rely on color coding can be accelerated by a factor of $O^*(e^k)$ by applying multi-linear monomial detection. For k-PATH, one obtains $O^*(2^k)$.
- Is it possible to solve faster by going to more exotic algebras? No! Koutis, Williams, ICALP '09.

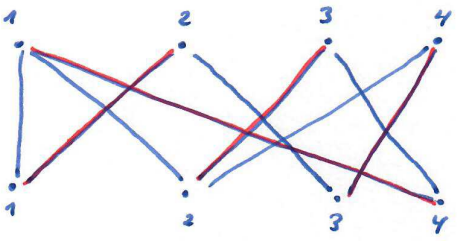
30.3 Further Algebraic Tools

PERFECT MATCHING

Given: Pairs $T \subseteq X \times Y$ with $|X| = |Y| = n$.

Question: $\exists T' \subseteq T$: $|T'| = n$ and
T' covers all vertices.

Illustration:



Lemma:
- Solving PERFECT MATCHING is in P.
- Counting the number of perfect matchings is #P-complete (as hard as counting the number of solutions for any NP-complete problem).

Computing the number of perfect matchings modulo 2 is easier.

Definition:
The incidence matrix of a bipartite graph over $X = [1,n]$ and $Y = [1,n]$ is defined by

$$A(i,j) = \begin{cases} 1, & \text{if } i \in X,\ j \in Y, \text{ and both are connected,} \\ 0, & \text{otherwise.} \end{cases}$$

Let N denote the number of perfect matchings of the given instance.
Let A be the incidence matrix.

Lemma (Matching Lemma):
- $N \bmod 2 = \det(A) \bmod 2$.
- The determinant can be computed with a polynomial number of arithmetic operations.

Remark:

- The Matching Lemma can be generalized to multigraphs where the edges are labelled by monomials.
- One can assign fingerprints such that not all monomials in the determinant have unique fingerprints. Instead, many graph structures that are <u>not</u> matchings come in pairs and in this way naturally cancel out.
- So computing modulo 2 is a resource, not a nuisance.