Title: **Detecting Multilinear Monomials with Algebraic Fingerprinting**

Author: Onni Miettinen

Advisor: Augusto Modanese

Date: 12.2.2023

TABLE OF CONTENTS

ABSTRACT

1 Introduction x

1.1 Preliminaries x

1.2 Algebraization of combinatorial problems x

2 Detecting multilinear monomials x

2.1 Dynamically expanding the polynomial x

2.2 Color coding for faster evaluation x

2.3 Color coding with matrices x

2.4 Fingerprinting monomials x

3 Implementing algebraic fingerprinting x

3.1 Problem A x

3.2 Problem B x

4 Conclusion x

References x

# Introduction

In recent years, there have been rapid advances in the algorithms for combinatorial problems. After several decades of no progress, Björklund et al. (2017) managed to design an algorithm that solves the Hamiltonian problem, i.e., finding whether a given graph contains a path which contains all vertices, significantly faster than the previously fastest algorithms. This development resulted in faster algorithms for several other problems. All this progress has been due to the development in algebraic methods for solving combinatorial problems, namely the development in algebraic fingerprinting, which will be the focus of this paper.

## Preliminaries

Before discussing the algebraization of problems, it is necessary to recall basic algebraic concepts. In this section, we will define an algebraic ring and field, and a few concepts regarding them.

An algebraic ring is a triple , where *R* is a set, and + and ∙ are binary operations over *R* called addition and multiplication, respectively, with both being associative and addition also being commutative. Multiplication has left and right distributiveness over addition, i.e., . The additive identity and multiplicative identity are notated with ***0*** and ***1***, respectively, and are defined with the following equations: . Moreover, every element of a ring *R* has an additive inverse, i.e., **.**

An element *u* of a ring *R* is a unit, if it has a multiplicative inverse, i.e., . If a ring has a commutative multiplication, every non-zero element is a unit, and , it is called a field. The characteristic for a field or ring *R* is defined with the following: . For example, for an element *u* of a field with characteristic *2*, the following equations hold: [ref].

## Algebraization of combinatorial problems

Designing an algebraic algorithm for a problem begins by the algebraization of that problem, i.e., transforming the original combinatorial question into a question regarding some algebraic property of some algebraic object built from the elements of the original problem. The recent success has been found by using multivariate polynomials; we construct a multivariate polynomial such that it encodes all combinations of the combinatorial problem, the solutions and non-solutions, as monomials, with the solutions being multilinear. Thus, finding whether a solution exists to the combinatorial problem has been transformed into finding whether the constructed polynomial contains a multilinear monomial.

Koutis and Williams (2015) showcased an example of this in their article with the parameterized problem of k-3D matching. From a set of triples, this problem asks for a *k*-matching, i.e., a subset of the input with disjoin elements. This combinatorial problem can be transformed into an algebraic form by viewing each element of each input triple as a unique variable, transforming the input triples into multivariate monomials by taking the product of the elements, summing those monomials together, and constructing the final polynomial by raising the sum to the power of *k*. Thus, the problem of finding a *k*-matching transforms into a problem of finding a multilinear monomial within the constructed polynomial. A decision problem version of this can be answered by finding whether a multilinear monomial exists.

The detection of multilinear monomials is a very important issue since a fast solution will make every algorithm, that utilizes this algebraization, faster. However, when the problem domain is large, with *n* variables in an *N*-degree polynomial, the number of possible monomials is . This motivates the detection of multilinear monomials without fully expanding the polynomial into a sum of monomials, which will be the topic of the next section.

# Detecting multilinear monomials

To tackle the issue with polynomials of large degree and dimension, we focus on an important observation: only the multilinear monomials in the polynomial are important, and everything else can be discarded. This implies that while expanding the polynomial, i.e., multiplying the terms with each other according to the degree of the polynomial and forming the eventual sum of products – the expanded polynomial, a monomial can be discarded as soon as one of its variables is squared. Thus, an algorithm utilizing dynamic programming can be designed for expanding the polynomial.

## Dynamic expansion of the polynomial

A dynamic programming algebra can be utilized for computing the sum-of-monomials form for the polynomial. First, it is necessary to set a condition which states that any variable squared equals to zero. This allows the dynamic programming algebra to essentially discard any non-multilinear monomial before it is even completely formed, which results in less arithmetic operations overall for computing the expansion. Therefore, dynamically expanding the polynomial saves computing time.

With the condition of identically equating squared variables to zero, we can formulate an algebraic form for the existence of a multilinear monomial: a multilinear monomial exists in the polynomial if the polynomial is not identically zero. This, of course, implies a solution to any combinatorial problem that can be algebraized into a multilinear monomial detection problem.

TODO: explain the design for dynamic programming algebra further

## Color coding and randomized evaluation

# Utilizing algebraic fingerprints

## Hamiltonicity

Hamiltonicity asks whether a graph contains a Hamiltonian path, i.e., a path that contains every vertex once. For a long period of time, the fastest algorithm for undirected Hamiltonicity ran in O(2^n) time, where n was the number of vertices in the input graph.

## K-path

Currently, the fastest algorithms for the k-path problem utilize algebraization along with algebraic fingerprints. Koutis [ref] introduced an algorithm that runs in O\*(2^k) time, which was a significant improvement on the earlier O\*((2e)^k) algorithm that was based on color coding

## Filtering non-solution monomials with fingerprints

K-path (narrow sieves for parameterized packings and paths), Björklund managed to form an algebraization and auxiliary fingerprints such that some non-solution monomials cancelled each other out. Non-path terms had a non-path pair, which in field of characteristic 2 cancels each other. Rest of the non-solution terms could be filtered out with inclusion-exclusion sieves.

**REFERENCES**

Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto. *Narrow sieves for parameterized paths and packings*. Journal of Computer and System Sciences. p. 119-139. 2017

Ioannis Koutis, Ryan Williams. *Algebraic Fingerprints for faster algorithms*. Annual ACM-SIAM Symposium on Discrete Algorithms. CA, USA. p. 1671-1680. 2015

**REFERENCES**

Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto. *Narrow sieves for parameterized paths and packings*. Journal of Computer and System Sciences. p. 119-139. 2017

Ioannis Koutis, Ryan Williams. *Algebraic Fingerprints for faster algorithms*. Annual ACM-SIAM Symposium on Discrete Algorithms. CA, USA. p. 1671-1680. 2015