A heuristic technique for decomposing multisets of non-negative integers according to the Minkowski sum

Luciano Margara

Department of Computer Science and Engineering, University of Bologna, Italy

received 2nd Aug. 2022, revised 10th Oct. 2022, accepted 25th Oct. 2022.

We study the following problem. Given a multiset M of non-negative integers, decide whether there exist and, in the positive case, compute two non-trivial multisets whose Minkowski sum is equal to M. The Minkowski sum of two multisets A and B is a multiset containing all possible sums of any element of A and any element of B. This problem was proved to be NP-complete when multisets are replaced by sets. This version of the problem is strictly related to the factorization of boolean polynomials that turns out to be NP-complete as well. When multisets are considered, the problem is equivalent to the factorization of polynomials with non-negative integer coefficients. The computational complexity of both these problems is still unknown.

The main contribution of this paper is a heuristic technique for decomposing multisets of non-negative integers. Experimental results show that our heuristic decomposes multisets of hundreds of elements within seconds, independently of the magnitude of numbers belonging to the multisets. Our heuristic can also be used for factoring polynomials in $\mathbb{N}[x]$. We show that, when the degree of the polynomials gets larger, our technique is much faster than the state-of-the-art algorithms implemented in commercial software like Mathematica and MatLab.

Keywords: multisets, polynomials, decomposition, heuristics

1 Introduction

The idea of decomposing a mathematical object into the sum (product, or other operations) of smaller ones is definitely not new. A huge literature has been devoted to the factorization of numbers, polynomials, matrices, graphs and many other mathematical objects, including sets and multisets. The basic idea behind factorization is decomposing a complex object into smaller and easier to analyze pieces. Properties satisfied by each piece might shed some light on the properties satisfied by the entire object. As an example, from irreducible factors of a polynomial, we can recover valuable information about its roots. In this paper, we study the decomposition of multisets of non-negative integers according to the Minkowski sum. Multisets are an extension of the notion of sets where, basically, multiple copies of the same element are allowed. The Minkowski sum is a binary operation that can be applied both to sets and multisets. The Minkowski sum of two multisets A and B is a multiset containing all possible sums of any element of A and any element of B.

ISSN 1365–8050

Given a multiset M of non-negative integers, the decomposition problem asks for computing two non-trivial multisets whose Minkowski sum is equal to M. Multisets theory have applications in many fields Singh et al. (2007), e.g., in combinatorics Anderson (2002); Stanley (2011); Stanley and Fomin (1999), in the theory of relational databases Grumbach and Milo (1996); Henglein et al. (2022); Lamperti et al. (2000), in multigraphs theory DeVos et al. (2013); Dudek et al. (2013) and in computational geometry Emiris et al. (2017). The problem of decomposing multisets of non-negative integers is strictly related to the problem of factoring univariate polynomials with non-negative coefficients (see Section 2.1 for details). Even if this problem arises in a very natural way in a number of different theoretical and practical contexts, it has not been thoroughly studied (see for example Brunotte (2013); Campanini and Facchini (2019); Van de Woestijne (2012)) and its computational complexity is still unknown. To our knowledge, no polynomial time algorithm nor an NP-completeness proof exists. When multisets are replaced by sets, the decomposition problem was proved to be NP-complete Kim and Roush (2005). Other variants of the Minkowski sum decomposition problem have been studied. As an example, in Gao and Lauder (2001) the authors study the Minkowski decomposition of integral convex polytopes proving that the decisional version of this problem is again NP-complete.

The main contribution of this paper is a heuristic technique for decomposing multisets of non-negative integers which, in turn, can be applied to factoring polynomials with non-negative coefficients.

The idea behind our algorithm is to transform the decomposition problem in an optimization problem by introducing a score function for candidate solutions. A candidate solution is an approximation of a solution. The score function measures the quality of candidate solutions, i.e., the similarity to the actual solution (not necessarily unique). The score function reaches its maximum (whose value is known in advance) only at a solution for the problem. Our algorithm starts from a randomly generated candidate solution s_0 and iteratively improves it until it finds a local optimum candidate solution s_k according to the score function. If s_k reaches the maximum score the algorithms terminates, otherwise it starts over from another initial candidate solution computed starting from s_k. The maximum number of iterations is bounded by a predetermined threshold.

We extensively tested our algorithm over randomly generated instances of different size and structure. Experimental results (see Section 4 and Tables in Appendix A and B) show that after a small number of iterations our algorithm almost always finds a solution.

As far as polynomial with non-negative coefficients factorization is concerned, no efficient and specifically designed algorithms are known. A possible natural strategy to solve this problem might consist of factoring the polynomial in $\mathbb{Z}[x]$ (this can be done in polynomial time) and then suitably grouping factors in $\mathbb{Z}[x]$ in order to get factors in $\mathbb{N}[x]$. Unfortunately, there exists no efficient algorithm to perform the grouping of factors whose number can be, in general, exponentially large. In our opinion, this is an interesting problem in itself. Since decomposing multisets of non-negative integers is equivalent (under some conditions we will discuss in Section 2.1) to the problem of factoring polynomials in $\mathbb{N}[x]$, the alternative strategy might also be used for decomposing multisets. In Section 5 we make a comparison between our algorithm and the alternative strategy depicted above unrealistically assuming that the grouping of factors can be computed for free. We used built-in functions provided in Wolfram Mathematica language for integer polynomials factorization (similar results have been found using MatLab).

Experimental results clearly show (see Tables 13,14 and 15 in Appendix B) that, when the degree of polynomials increases, our technique is much faster than going through factoring. Reversing the line of reasoning, i.e., using multisets decomposition techniques for factoring polynomials in $\mathbb{N}[x]$, our heuristics becomes a serious candidate to be the first effective method for factoring polynomials with non-negative coefficients.

The rest of this paper is organized as follows. In Section 2 we give basic definitions and known results. In Section 3 we describe our heuristics and we provide its pseudocode. In Section 4 we show experimental results. In Section 5 we make a comparison between our algorithm and an alternative strategy for decomposing multisets based on integer polynomial factorization. Section 6 contains conclusions and some ideas for further works. Appendices A and B contain tables with experimental data.

2 Definitions and Known Results

Let $\mathbb{Z}$ be the set of integers and $\mathbb{Z}[x]$ be the sets of univariate polynomials with coefficients in $\mathbb{Z}$. Let $\mathbb{N}$ be the set of non-negative integers and $\mathbb{N}[x]$ be the sets of univariate polynomials with coefficients in $\mathbb{N}$.

Multisets are an extension of the concept of sets. While a set can contain only one occurrence of any given element, a multiset may contain multiple occurrences of the same element. To distinguish multisets from sets, we will represent multisets by using double braces.

As an example $M = \{\{2, 2, 3, 3, 5, 5, 5, 5, 5, 6, 8, 8\}\}$ is a multiset. Given a multiset M we denote by $\mu(x, M)$ the number of occurrences (possibly 0) of the element x in M. Sometimes we will represent a multiset M as a set of pairs $(element, \mu(element, M)$. With this notation, the above multiset can be written as $M = \{(2, 2), (3, 2), (5, 5), (6, 1), (8, 2)\}$. In what follows, we will consider sets and multisets of numbers. This enable us to define a binary operation on them (denoted by the symbol $\oplus$) sometimes called Minkowski sum. We will use the symbol $\oplus$ both for sets and multisets sum inferring the type of operation from the type of operands.

Definition 1 (Minkowski Set Sum)**.** *The Minkowski sum of two sets A and B is a set defined as follows.*

$$A \oplus B = \{a + b : \ a \in A \text{ and } b \in B\}$$

Example 1. *Example of set sum. Let $A = \{0, 1, 3\}$ and $B = \{2, 5\}$. Then $A \oplus B = \{2, 3, 5, 6, 8\}$. Since we are working with sets, the multiplicity of 5 in $A \oplus B$ is 1 even if 5 can be obtained both as $0 + 5$ and $3 + 2$.*

Definition 2 (Minkowski Multiset Sum)**.** *The Minkowski sum of two multisets A and B is a multiset given by*

$$A \oplus B = \{\{a + b : \ a \in A \text{ and } b \in B\}\}$$

Example 2. *Examples of multiset sum.*
Let $A = \{\{0, 1, 3\}\}$ and $B = \{\{2, 5\}\}$. Then $A \oplus B = \{\{2, 3, 5, 5, 6, 8\}\}$.
Let $A = \{\{0, 1, 3, 3\}\}$ and $B = \{\{2, 2, 5\}\}$. Then$A \oplus B = \{\{2, 2, 3, 3, 5, 5, 5, 5, 5, 6, 8, 8\}\}$.

The identity element with respect to the set sum is $\{0\}$ and the identity element with respect to the multiset sum is $\{\{0\}\}$. A multiset A is contained in a multiset B ($A \subseteq B$) if and only if

$$\forall x \in A : \quad x \in B \text{ and } \mu(x, A) \le \mu(x, B) \tag{1}$$

We also define the multiset difference operation (denoted by the $\setminus$ symbol) as follows.

$$A \setminus B = \{(x, m_x) : \ x \in A \text{ and } m_x = \max(\mu(x, A) - \mu(x, B), 0)\} \tag{2}$$

As an example, $\{\{2, 2, 3, 3, 5, 6, 8, 8\}\} \setminus \{\{2, 3, 3, 3, 5, 9\}\} = \{\{2, 6, 8, 8\}\}$. We now introduce the notion of reducible multisets (sets) of non-negative integers.

Definition 3 (Reducible multiset (set))**.** *A multiset (set) M of non-negative integers is reducible if and only if there exist two multisets (sets) A and B, both of them different from the identity element, such that $M = A \oplus B$.*

A multiset (set) M of non-negative integers is irreducible (sometimes called prime) if and only if it is not reducible. We are now ready to state the following two problems.

Definition 4 (SET-RED)**.** *Given a set S of non-negative integers, decide whether S is reducible or not.*

Definition 5 (MULTISET-RED)**.** *Given a multiset M of non-negative integers, decide whether M is reducible or not.*

The following result was proved in Gao and Lauder (2001).

Theorem 1. SET-RED *is NP-complete.*

Unlike SET-RED, the computational complexity of MULTISET-RED is, to our knowledge, still unknown. This leads us to state the following open question.

Question 1. *Is* MULTISET-RED *NP-complete ?*

Even if we have defined SET-RED and MULTISET-RED in their decisional version, in the rest of this paper we will refer to them (with a little abuse of notation) as constructive problems, i.e, the problem of effectively computing two multisets (sets) whose Minkowski sum is equal to the multiset (set) received as input.

In the next example we show that the irreducible factorization of non-negative integer multisets is not unique. This makes the problem of factoring multisets even harder, if possible.

Example 3. *Let $M = \{\{0, 1, 2, 3, 4, 5\}\}$. Then*

$$\begin{aligned} M &= \{\{0, 1\}\} \oplus \{\{0, 2, 4\}\} \\ &= \{\{0, 3\}\} \oplus \{\{0, 1, 2\}\}. \end{aligned}$$

Multisets $\{\{0, 1\}\}$, $\{\{0, 2, 4\}\}$, $\{\{0, 3\}\}$ and $\{\{0, 1, 2\}\}$ are irreducible.

2.1 Multisets decomposition and polynomials factorization

One of the most studied problem in computer algebra is the problem of factoring polynomials. A huge literature has been devoted to the factorization of polynomials (without claim of exhaustiveness see Hoeij (2002); Lenstra et al. (1982); Kaltofen (1992)). The first polynomial factorization algorithm was published by Theodor Von Schubert in 1793 Schubert (1793). Since then, dozens of papers on the computational complexity of polynomial factorization have been published. In 1982, Arjen K. Lenstra, Hendric W. Lenstra, and László Lovász Lenstra et al. (1982) published the first polynomial time algorithm for factoring polynomials over $\mathbb{Q}$ and then over $\mathbb{Z}$.

The problem of factoring polynomials over a ring can be, in a sense, labeled as "well studied" and "efficiently solved". The same cannot be said when rings are replaced by semirings (e.g. the natural numbers). Unlike the case of factoring polynomials over rings, the problem of factoring polynomials over semirings has received far less attention, there are far fewer known results and many interesting unanswered questions. One of them is the following.

Question 2 ($\mathbb{N}$-POLY-RED)**.** *Given a polynomial $p(x) \in \mathbb{N}[x]$, decide whether $p(x)$ is reducible in $\mathbb{N}[x]$.*

As far as we know, for the $\mathbb{N}$-POLY-RED problem, there are neither polynomial algorithms to solve it nor proofs of NP-completeness. $\mathbb{N}$-POLY-RED problem is strictly related to the MULTISET-RED problem.

To any given polynomial $p(x) \in \mathbb{N}[x]$ it is possible to associate a multiset as follows. Let $p = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ be any element of $\mathbb{N}[x]$. We define the multiset

$$Multiset(p) = \{\{\overbrace{0,\ldots,0}^{a_0},\ldots,\overbrace{i,\ldots,i}^{a_i},\ldots,\overbrace{n,\ldots,n}^{a_n}\}\} \tag{3}$$

On the other hand, we can associate to any multiset

$$M = \{\{\overbrace{n_1,\ldots,n_1}^{m_1},\overbrace{n_2,\ldots,n_2}^{m_2},\ldots,\overbrace{n_d,\ldots,n_d}^{m_d}\}\}$$

the polynomial

$$Polynomial(M) = m_1x^{n_1} + m_2x^{n_2} + \cdots + m_dx^{n_d} \tag{4}$$

It is not difficult to verify that
- $Polynomial(Multiset(p)) = p$ and $Multiset(Polynomial(M)) = M$
- $Multiset(p\,q) = Multiset(p) \oplus Multiset(q)$ and
- $Polynomial(A \oplus B) = Polynomial(A)\,Polynomial(B)$

As a consequence of these properties we have that
- M is an irreducible multiset of non-negative integers if and only if $Polynomial(M)$ is an irreducible polynomial over $\mathbb{N}[x]$ and
- p is an irreducible polynomial over $\mathbb{N}[x]$ if and only if $Multiset(p)$ is an irreducible multiset of non-negative integers.

Unfortunately, in the general case, the size of $Multiset(p)$ may be exponentially larger than the size of p. This prevents us from readily translating computational complexity results for MULTISET-RED into equivalent results for $\mathbb{N}$-POLY-RED and viceversa.

Taking advantage of Example 3 we show that the irreducible factorization of polynomials in $\mathbb{N}[x]$ is not unique.

Example 4. *Let $p(x) = 1 + x + x^2 + x^3 + x^4 + x^5$. The complete factorization of $p(x)$ in $\mathbb{Z}[x]$ is $p(x) = (1+x)(1-x+x^2)(1+x+x^2)$. Since $(1+x)(1-x+x^2) \in \mathbb{N}[x]$ and $(1-x+x^2)(1+x+x^2) \in \mathbb{N}[x]$, then we have two distinct factorizations of $p(x)$ in $\mathbb{N}[x]$.*

$$\begin{aligned} p(x) &= (1+x)(1+x^2+x^4) \\ &= (1+x^3)(1+x+x^2) \end{aligned}$$

3 The Heuristics

In this section we provide a complete description of our heuristics by using pseudocode (for details see pages from 20 to 22 in Cormen et al. (2009)).

Given a multiset M of n non-negative integer numbers, a candidate solution for M is any multiset A ($A \neq \{\{0\}\}$) of cardinality m such that $A \subseteq M$ and m divides n. A candidate solution A for M is also a solution for M if and only if there exists another candidate solution B ($B \neq \{\{0\}\}$) for M such that $M = A \oplus B$. Given a candidate solution A for M, deciding whether A is also a solution for M can be done in polynomial time. Given a solution A for M, computing B such that $M = A \oplus B$ can be done in polynomial time.

Our heuristics starts from an initial candidate solution of a given cardinality and iteratively improves it (according to a given score function) until it finds a solution. The cardinality m of the initial candidate solution is unknown in advance but must divide the cardinality of M. For computing an actual decomposition of a multiset M of cardinality n we have to run our algorithm on all possible factors f of n with $f \leq \sqrt{n}$. We are aware that this leads to an overhead of computation, but luckily, the number of factors of any positive integer n (not exceeding $\sqrt{n}$) is very small if compared to n. For every positive integer n, with $100 \leq n \leq 100.000$, we computed its number of factors divided by n. It turns out that the average of these ratios is 0.00025 and the maximum is 0.058 (higher values are obtained for small numbers). For these reasons, in what follows, we will assume that the target cardinality of solutions is known.

We now give the pseudocode of each function used in our heuristics and a short explanation on how it works.

```
INITIALSOLUTION(M, n)
  m = ROUND(M.length/2)
  M = SORT(M)
  M = M[1..m]
  M = RANDOMSAMPLE(M, n)
  return M
```

INITIALSOLUTION takes as input a multiset M and a non-negative integer n that divides the cardinality of M and returns a candidate solution of cardinality n.

```
SCORE(M, S)
  // invariant: S[1] = 0, S ⊆ M and S.length divides M.length
  col = S.length
  row = M.length/col // Let mat be an row × col matrix whose entries are set to 0
  r = M \ S
  // first row of mat gets S
  score = col
  for i = 2 to row
      w = MIN(r)
      r = r \ {w}
      score = score + 1
      // mat[row, 1] = w
      for j = 2 to col
          c = w + S[j]
          if c ∈ r
              r = r \ {c}
              score = score + 1
              // mat[row, col] = c
          else return score
  return score
```

SCORE takes as input a multiset M and a candidate solution S for M and returns a positive integer measuring the quality of S. SCORE(M, S) ranges from length of S (lowest quality) to length of M (highest quality). If SCORE$(M, S) =$ length of M then S is a solution for M.

To better understand how SCORE works, we describe its behavior on the following example. Let $A = \{\{0, 1, 3, 3\}\}$, $B = \{\{0, 2, 2, 6\}\}$, and

$$M = A \oplus B = \{\{0, 1, 2, 2, 3, 3, 3, 3, 5, 5, 5, 5, 6, 7, 9, 9\}\}$$

Assume now to run SCORE(M, B). Since B is a solution for M, SCORE(M, B) returns 16, i.e., the length of M. The matrix mat described (but not computed) at lines 6,8,14 and 20 would be

$$mat = \begin{bmatrix} 0 & 2 & 2 & 6 \\ 1 & 3 & 3 & 7 \\ 3 & 5 & 5 & 9 \\ 3 & 5 & 5 & 9 \end{bmatrix}$$

and the elements of mat would give exactly the multiset M.

Assume now to run SCORE(M, C). Where $C = \{\{0, 1, 2, 6\}\}$ is a candidate solution but not a solution. SCORE(M, C) returns 6. The matrix mat would now have the form

$$mat = \begin{bmatrix} 0 & 1 & 2 & 6 \\ 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The element at row 2 and column 3 $(2 + 2 = 4)$ in mat cannot be found in M (note that we have already removed $0, 1, 2, 6, 2$ and 3 from M) and then SCORE(M, C) stops at line 21 returning 6, i.e., the number of elements correctly placed in mat until that moment.

Last case. Assume to run SCORE(M, C). Where $C = \{\{0, 2, 2, 5\}\}$ is again a candidate solution but not a solution. SCORE(M, C) returns 11. The matrix mat would have now the form

$$mat = \begin{bmatrix} 0 & 2 & 2 & 5 \\ 1 & 3 & 3 & 6 \\ 3 & 5 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The element at row 3 and column 4 $(3 + 5 = 8)$ in mat cannot be found in M and then SCORE(M, C) stops at line 21 returning 11, i.e., the number of elements correctly placed in mat until that moment.

NEIGHBORSEARCH(M, S)

```
1  // invariant: S[1] = 0 and S.length divides M.length
2  initial_score = SCORE(M, S)
3  alternatives = DELETEDUPLICATES(M \ S)
4  for i = 2 to S.length
5      for j = 1 to alternatives.length
6          temp = S[i]
7          S[i] = alternatives[j]
8          new_score = SCORE(M, S)
9          if new_score > initial_score
10             return (new_score, S)
11         else S[i] = temp
12 return (initial_score, S)
```

NEIGHBORSEARCH takes as input a multiset M and a candidate solution S for M and returns a candidate solution N in the neighborhood of S such that SCORE$(M, N) >$ SCORE(M, S), if any. Returns S, otherwise.

Given a multiset M and a candidate solution S for M, a neighbor of S is any candidate solution for M differing from S for exactly 1 element. To speed up the process, NEIGHBORSEARCH returns (line 11) the first improved candidate solution found.

FINDLOCALOPT(M, S)

```
// invariant: S[1] = 0
n = M.length
current_score = SCORE(M, S)
while TRUE
    (score, S) = NEIGHBORSEARCH(M, S)
    if score == n
        return (TRUE, S)
    if score == current_score
        return (FALSE, S)
    current_score = score
```

FINDLOCALOPT takes as input a multiset M and a candidate solution S for M and returns a candidate solution N with the property of being the best candidate solution in its neighbor, i.e., a local optimum. To accomplish this task, FINDLOCALOPT keeps on calling NEIGHBORSEARCH on improved solutions until no more improvement is found. Note that the candidate solution N produced by FINDLOCALOPT is not guaranteed to be a solution.

```
ITERATEDSEARCH(M, m, iterations)
// invariant: m divides M.length
current_solution = INITIALSOLUTION(M, m)
for i = 1 to iterations
    (found, S) = FINDLOCALOPT(M, current_solution)
    if found
        return S
    current_solution = NEWINITIALSOLUTION(M, current_solution)
    // note that current_solution contains 0
return solution not found
```

ITERATEDSEARCH takes as input a multiset M, an integer $m > 1$ dividing the cardinality of M and an upper bound on the number of iterations and returns a solution of cardinality m, if found. ITERATEDSEARCH keeps on calling FINDLOCALOPT with different initial candidate solutions (computed by NEWINITIALSOLUTION) until a solution is found or the maximum number of iterations is exceeded.

```
NEWINITIALSOLUTION(M, S)
// invariant: S[1] = 0, all the elements of S are in M and S.length divides M.length
col = S.length
row = M.length/col
// Let mat be an row × col matrix whose entries are set to 0
R = M \ S
// first row of mat gets S
new_set = S
for i = 2 to row
    w = MIN(R)
    R = R \ {w}
    new_set = new_set ⋃{w}
    // mat[row, 1] = w
    for j = 2 to col
        c = w + S[j]
        if c ∈ R
            r = R \ {c}
            // mat[row, col] = c
        else return RANDOMSAMPLE(new_set, col)
            // RANDOMSAMPLE(new_set, col) must contain 0
return RANDOMSAMPLE(new_set, col)
```

NEWINITIALSOLUTION takes as input a multiset M and a candidate solution S for M and returns a new initial candidate solution. To better understand how NEWINITIALSOLUTION works, we show its behavior on an example. Let $A = \{\{0, 1, 3, 3\}\}$, $B = \{\{0, 2, 2, 6\}\}$, and

$$M = A \oplus B = \{\{0, 1, 2, 2, 3, 3, 3, 3, 5, 5, 5, 5, 6, 7, 9, 9\}\}$$

Assume to run NEWINITIALSOLUTION(M, C). Where $C = \{\{0, 2, 2, 5\}\}$ is a candidate solution but not a solution. The matrix mat, if computed, would have the form

$$mat = \begin{bmatrix} 0 & 2 & 2 & 5 \\ 1 & 3 & 3 & 6 \\ 3 & 5 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

NEWINITIALSOLUTION(M, C) stops at line 20 returning $\{\{0, 2, 2, 5, 1, 3\}\}$, i.e., the union of the first row of mat and the initial part (first 3 elements) of the first column of mat. Experimental results clearly show that solutions to the problem contains with high probability elements placed in the first row or in the first column of the matrix mat associated to the local optimum candidate solution.

4 Experimental results

We tested our algorithm on an iMac equipped with a 4.2 GHz Intel Core $i7$ quad-core processor and 32 GB RAM (2400 MHz DDR4). Operating System: macOS Monterey Version 12.2.1. Our algorithm has been implemented in Wolfram Mathematica language (Version 12). To make the code more readable even to those unfamiliar with the Mathematica language, we decided to describe it providing a pseudocode version (see Section 3).

Our algorithm has been extensively tested over instances (multisets of non-negative integers) of different size and structure. Instances depend on two parameters, namely $structure$ and $range$, and have been generated according to the following procedure.

INSTANCEGENERATION$(structure, range)$

1 $inst = \{\{0\}\}$
2 **for** $i = 1$ **to** $structure.length$
3   Let M be a multiset with the following properties:
4   - cardinality of M is equal to $structure[i]$
5   - M contains at least one element equal to 0
6   - each element of M is randomly chosen in the interval $[0\,..\,range]$
7     $inst = inst \oplus M$
8 **return** $inst$

The parameter $structure$ is a list of positive integers representing the cardinalities of the multisets that, once summed together, produce the instance. The parameter $range$ represents an upper bound on the numbers in the multisets (see line 6 of INSTANCEGENERATION). As an example, the instance produced by INSTANCEGENERATION$(\{2,2,3\}, 10)$ is a multiset of cardinality $12 = 2 \times 2 \times 3$ obtained by summing up 3 randomly generated multisets of cardinality $2, 2$ and 3, respectively. Each element of the 3 multisets is randomly chosen from the set $\{0, 1, \ldots, 10\}$. We only consider multisets containing at least one element equal to zero. In fact, any multiset M that does not contain 0, i.e., $\mu(0, M) = 0$, can be always decomposed as $\{\{min(M)\}\} \oplus M'$ where M' is a multiset obtained from M subtracting to each element $min(M)$. As an example, $\{\{2, 4, 3, 4, 3, 5\}\} = \{\{2\}\} \oplus \{\{0, 2, 1, 2, 1, 3\}\}$.

For each $structure$ and $range$, we tested our algorithms on a large number of instances collecting results in Tables 1 to 12 in Appendix A.

Columns of Tables contain the following data.

1. $Size$: size of the input, i.e., cardinality of the considered multiset

2. $Structure$: structure of the considered multiset
3. $Success$: percentage of runs for which a solution is found
4. $Iterations$: Average number of iterations for any given $structure$
5. $Time$: Average running time for any given $structure$
6. $Time/Iter$: $Time$ divided by $Iterations$
7. $Time/Size$: $Time$ divided by $Size$ We investigated the performance of our algorithm in different scenarios. Number of duplicates. We tested our algorithm with two different values of the parameter $range$. Namely, $range = 5$ and $range = 10000$. In the case of $range = 5$, multisets contain a large number of duplicates, while in the case of $range = 10000$ duplicates are very rare.

with 3 different type of structures $\{n, n\}$, $\{2\,n, n\}$ and $\{n, \ldots, n\}$.
- $\{n, n\}$: sum of two multisets with the same cardinality;
- $\{2\,n, n\}$: sum of two multisets with different cardinalities (one half of the other);
- $\{n, \ldots, n\}$: sum of k multisets with the same cardinality (denoted by $\{n\}^k$).

We now give some reading keys and interpretations of experimental data collected in Tables 1 to 12 in Appendix A.

ITERATEDSEARCH finds a solution most of the time. Leaving unbounded the maximum number of allowed iterations, ITERATEDSEARCH always finds a solution. From a practical point of view, leaving unbounded the number of iterations prevents the algorithm to recognize irreducible multisets. In our tests we set the maximum number of iterations equal to 100. Even in this case, ITERATEDSEARCH is able to find a solution approximately 999 times out of 1000.

Multisets with many duplicates approximately takes the same amount of time to decompose with respect to multisets with a small number of duplicates. The presence of many duplicates forces the heuristics to go through a larger number of iterations to find a solution but single iterations are much faster. With many duplicates, the behavior of ITERATEDSEARCH is less regular in terms of running times and distribution of failures.

Multisets obtained summing up many small multisets are much easier to decompose with respect to multisets obtained summing up 2 large multisets. As an example, a multiset with structure $\{2\}^{15}$ and size 32768 takes approximately the same time (last row of Table 3) of a multiset with structure $\{20, 20\}$ and size 400 (last row of Table 2). For multisets obtained summing up many small multisets, the average number of iterations is very close to 1.

5 Polynomial Factorization vs Iterated Search

An alternative strategy for decomposing a multiset of non-negative integers (or, equivalently, an intuitive way of factoring a polynomial in $\mathbb{N}[x]$) might be the following.

ALTERNATIVESTRATEGY(M)

1 // M is a multiset of non-negative integers
2 p = POLYNOMIAL(M)
3 *fl* = FACTORLIST(p)
4 (P_1, P_2) = GROUP(*fl*)
5 **return** $(Multiset(P_1), Multiset(P_2))$

Line 2 computes the polynomial p associated to the multiset M as shown in Equation (4). Line 3 computes the factor list fl of p. Line 4, using some unknown algorithm (it would be of some interest to find an algorithm for efficiently computing GROUP(*fl*)), computes a partition $P = \{P_1, P_2\}$ (if there exists one) of the factor list *fl* such that the product of all the polynomials in P_1 and the product of all the polynomials in P_2 have non-negative coefficients.

In what follows we will assume that the computational cost of Line 4 is zero. Table 13 to 15 compare running times of ITERATEDSEARCH and ALTERNATIVESTRATEGY for multisets with homogeneous $structure$ and increasing $ranges$.

For computing the factor list at Line 3 of ALTERNATIVESTRATEGY we make use of the function FACTORLIST provided by Mathematica Language (similar results are obtained by using the function FACTOR of MatLab).

Experimental results (see Tables 13,14 and 15) clearly show that the running time of ITERATEDSEARCH is independent of the magnitude of numbers in the multisets (exponents in the polynomials). ITERATEDSEARCH is much faster than ALTERNATIVESTRATEGY in the case of multisets containing large numbers and small multiplicity.

Doing the reverse path enable us to give a new technique for decomposing polynomials in $\mathbb{N}[x]$ based on ITERATEDSEARCH.

```
ℕ-POLYFACT(p)
// p ∈ ℕ[x]
M = MULTISET(p)
S = ITERATEDSEARCH(M) //
P = Polynomial(S)
return (Polynomial(S), p/P)
```

We end this section by giving a small multiset M of non-negative integers that ITERATEDSEARCH decomposes in 0.008 seconds. ALTERNATIVESTRATEGY (both using Mathematica and MatLab factorization primitives) called on the same multiset, after 24 hours of computation, was unable to find any solution.

$$A = \{\{0, 1249, 4270, 4324, 4852\}\}$$
$$B = \{\{0, 1705, 2250, 2267, 4390\}\}$$

$$\begin{aligned} M = A \oplus B \quad = \quad & \{\{0, 1249, 1705, 2250, 2267, 2954, 3499, 3516, 4270, 4324, 4390, 4852, 5639, \\ & 5975, 6029, 6520, 6537, 6557, 6574, 6591, 7102, 7119, 8660, 8714, 9242\}\} \end{aligned}$$

$$\begin{aligned} Polynomial(M) \quad = \quad & 1 + x^{1249} + x^{1705} + x^{2250} + x^{2267} + x^{2954} + x^{3499} + x^{3516} + x^{4270} + \\ & x^{4324} + x^{4390} + x^{4852} + x^{5639} + x^{5975} + x^{6029} + x^{6520} + x^{6537} + \\ & x^{6557} + x^{6574} + x^{6591} + x^{7102} + x^{7119} + x^{8660} + x^{8714} + x^{9242} \end{aligned}$$

6 Conclusions and further work

We have introduced and analyzed a heuristic technique for decomposing multisets of non-negative integers according to the Minkowski sum. Experimental results show that our technique allows to decompose quite large multisets (hundreds to thousands of elements depending on the instance structure) in seconds. Our technique can also be used to tackle the problem of factoring polynomials in $\mathbb{N}[x]$. Experimental results show that, when the size of exponents (elements of multisets) increases, our technique is much faster than state-of-the-art implementation of polynomial factoring algorithms over $\mathbb{Z}[x]$ that can be viewed as a preparatory step for factoring over $\mathbb{N}[x]$.

A natural extension of this work is replacing non-negative integers with more complex mathematical objects. It would be of some interest to investigate the case of d dimensional vectors of non-negative integers with $d > 1$. The problem of decomposing multisets of d dimensional vectors is strictly related to the problem of factoring multivariate polynomials with non-negative coefficients, but also to a number of problems arising, for example, in the field of computational geometry and seems to be more challenging than the 1 dimensional case.

It would be interesting to investigate whether the combination of the results obtained by using our algorithm on single components of the d dimensional object can be of any help for solving the global problem.

A Experimental Data Tables I

Tab. 1: $Range = 5$. Number of tested instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
9	$\{3, 3\}$	100	1	0.001	0.001	0.00011
16	$\{4, 4\}$	100	1.14	0.002	0.00175	0.00012
25	$\{5, 5\}$	100	1.4	0.008	0.00571	0.00032
36	$\{6, 6\}$	100	1.9	0.025	0.01316	0.00069
49	$\{7, 7\}$	100	2.6	0.061	0.02346	0.00124
64	$\{8, 8\}$	100	3	0.122	0.04067	0.00191
81	$\{9, 9\}$	100	4.22	0.242	0.05735	0.00299
100	$\{10, 10\}$	100	4.22	0.366	0.08673	0.00366
121	$\{11, 11\}$	100	7.06	0.934	0.13229	0.00772
144	$\{12, 12\}$	100	4.72	0.95	0.20127	0.0066
169	$\{13, 13\}$	100	11.7	2.728	0.23316	0.01614
196	$\{14, 14\}$	100	7.02	2.454	0.34957	0.01252
225	$\{15, 15\}$	100	7.14	3.298	0.4619	0.01466
256	$\{16, 16\}$	100	8.16	4.563	0.55919	0.01782
289	$\{17, 17\}$	100	10.72	8.151	0.76035	0.0282
324	$\{18, 18\}$	100	9.56	9.18	0.96025	0.02833
361	$\{19, 19\}$	99.9	11	12.168	1.10618	0.03371
400	$\{20, 20\}$	100	18.5	29.491	1.59411	0.07373

Tab. 2: $Range = 5$. Number of tested instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
18	$\{6, 3\}$	100	1.24	0.002	0.00161	0.00011
32	$\{8, 4\}$	100	1.92	0.011	0.00573	0.00034
50	$\{10, 5\}$	100	2.52	0.038	0.01508	0.00076
72	$\{12, 6\}$	100	3.1	0.104	0.03355	0.00144
98	$\{14, 7\}$	100	3.58	0.222	0.06201	0.00227
128	$\{16, 8\}$	100	4.68	0.452	0.09658	0.00353
162	$\{18, 9\}$	100	6.44	0.961	0.14922	0.00593
200	$\{20, 10\}$	100	9.22	1.865	0.20228	0.00932
242	$\{22, 11\}$	100	7.56	2.655	0.35119	0.01097
288	$\{24, 12\}$	100	9.4	4.161	0.44266	0.01445
338	$\{26, 13\}$	100	15.8	8.474	0.53633	0.02507
392	$\{28, 14\}$	99.9	12.1	10.56	0.87273	0.02694
450	$\{30, 15\}$	100	11.62	13.641	1.17392	0.03031

Tab. 3: $Range = 5$. Number of tested instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
8	$\{2\}^3$	100	1	0.001	0.001	0.00012
16	$\{2\}^4$	100	1	0.001	0.001	0.00006
32	$\{2\}^5$	100	1	0.001	0.001	0.00003
64	$\{2\}^6$	100	1	0.002	0.002	0.00003
128	$\{2\}^7$	100	1	0.004	0.004	0.00003
256	$\{2\}^8$	100	1	0.007	0.007	0.00003
512	$\{2\}^9$	100	1	0.014	0.014	0.00003
1024	$\{2\}^{10}$	100	1	0.037	0.037	0.00004
2048	$\{2\}^{11}$	100	1	0.11	0.11	0.00005
4096	$\{2\}^{12}$	100	1	0.375	0.375	0.00009
8192	$\{2\}^{13}$	100	1	1.15	1.15	0.00014
16384	$\{2\}^{14}$	100	1	4.708	4.708	0.00029
32768	$\{2\}^{15}$	100	1	18.625	18.625	0.00057

Tab. 4: $Range = 5$. Number of tested instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
27	$\{3\}^3$	100	1	0.003	0.003	0.00011
81	$\{3\}^4$	100	1.04	0.012	0.01154	0.00015
243	$\{3\}^5$	100	1	0.039	0.039	0.00016
729	$\{3\}^6$	100	1	0.175	0.175	0.00024
2187	$\{3\}^7$	100	1	1.088	1.088	0.0005
6561	$\{3\}^8$	100	1	6.646	6.646	0.00101
19683	$\{3\}^9$	100	1	60.155	60.155	0.00306

Tab. 5: $Range = 5$. Number of tested instances for each structure: 1000. For $Size = 16384$, due to time limits, we reduced the number of instances to 300.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
64	$\{4\}^3$	100	1.1	0.022	0.02	0.00034
256	$\{4\}^4$	100	1.02	0.14	0.13725	0.00055
1024	$\{4\}^5$	100	1.06	1.266	1.19434	0.00124
4096	$\{4\}^6$	100	1	11.377	11.377	0.00278
16384	$\{4\}^7$	100	1.24	366.325	295.423	0.02236

Tab. 6: $Range = 5$. Number of tested instances for each structure: 1000. For $Size = 15625$, due to time limits, we reduced the number of instances to 300.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
125	$\{5\}^3$	100	1.38	0.114	0.08261	0.00091
625	$\{5\}^4$	100	1.26	1.307	1.0373	0.00209
3125	$\{5\}^5$	100	1.12	23.818	21.2661	0.00762
15625	$\{5\}^6$	100	1.08	521.383	482.762	0.03337

Tab. 7: $Range = 10000$. Number of instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
9	$\{3,3\}$	100	1	0.001	0.001	0.00011
16	$\{4,4\}$	100	1	0.002	0.002	0.00012
25	$\{5,5\}$	100	1	0.008	0.008	0.00032
36	$\{6,6\}$	100	1	0.02	0.02	0.00056
49	$\{7,7\}$	100	1	0.05	0.05	0.00102
64	$\{8,8\}$	100	1	0.105	0.105	0.00164
81	$\{9,9\}$	100	1	0.199	0.199	0.00246
100	$\{10,10\}$	100	1	0.375	0.375	0.00375
121	$\{11,11\}$	100	1	0.606	0.606	0.00501
144	$\{12,12\}$	100	1	1.138	1.138	0.0079
169	$\{13,13\}$	100	1	1.815	1.815	0.01074
196	$\{14,14\}$	100	1	2.831	2.831	0.01444
225	$\{15,15\}$	100	1	4.064	4.064	0.01806
256	$\{16,16\}$	100	1	6.09	6.09	0.02379
289	$\{17,17\}$	100	1.4	10.515	7.51071	0.03638
324	$\{18,18\}$	100	1	13.469	13.469	0.04157
361	$\{19,19\}$	100	1	19.217	19.217	0.05323
400	$\{20,20\}$	100	1.02	27.122	26.5902	0.0678

Tab. 8: $Range = 10000$. Number of instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
18	$\{6, 3\}$	100	2.22	0.004	0.0018	0.00022
32	$\{8, 4\}$	100	1.78	0.012	0.00674	0.00038
50	$\{10, 5\}$	100	2.2	0.047	0.02136	0.00094
72	$\{12, 6\}$	100	1.76	0.096	0.05455	0.00133
98	$\{14, 7\}$	100	1.72	0.214	0.12442	0.00218
128	$\{16, 8\}$	100	1.96	0.488	0.24898	0.00381
162	$\{18, 9\}$	100	3.02	1.469	0.48642	0.00907
200	$\{20, 10\}$	99.9	6.14	6.141	100016	0.0307
242	$\{22, 11\}$	100	2.68	3.944	1.47164	0.0163
288	$\{24, 12\}$	100	1.64	4.777	2.9128	0.01659
338	$\{26, 13\}$	100	2.26	9.864	4.3646	0.02918
392	$\{28, 14\}$	100	2.06	15.012	7.28738	0.0383
450	$\{30, 15\}$	100	3.18	33.11	10.412	0.07358

Tab. 9: $Range = 10000$. Number of instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
8	$\{2\}^3$	100	1	0.001	0.001	0.00012
16	$\{2\}^4$	100	1	0.001	0.001	0.00006
32	$\{2\}^5$	100	1	0.001	0.001	0.00003
64	$\{2\}^6$	100	1	0.001	0.001	0.00002
128	$\{2\}^7$	100	1	0.002	0.002	0.00002
256	$\{2\}^8$	100	1	0.004	0.004	0.00002
512	$\{2\}^9$	100	1	0.009	0.009	0.00002
1024	$\{2\}^{10}$	100	1	0.024	0.024	0.00002
2048	$\{2\}^{11}$	100	1	0.07	0.07	0.00003
4096	$\{2\}^{12}$	100	1	0.228	0.228	0.00006
8192	$\{2\}^{13}$	100	1	0.811	0.811	0.0001
16384	$\{2\}^{14}$	100	1	2.984	2.984	0.00018
32768	$\{2\}^{15}$	100	1	11.708	11.708	0.00036

Tab. 10: $Range = 10000$. Number of instances for each structure: 1000.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
27	$\{3\}^3$	100	1	0.002	0.002	0.00007
81	$\{3\}^4$	100	1	0.008	0.008	0.0001
243	$\{3\}^5$	100	1	0.028	0.028	0.00012
729	$\{3\}^6$	100	1	0.139	0.139	0.00019
2187	$\{3\}^7$	100	1	0.916	0.916	0.00042
6561	$\{3\}^8$	100	1	7.047	7.047	0.00107
19683	$\{3\}^9$	100	1	72.214	72.214	0.00367

Tab. 11: $Range = 10000$. Number of instances for each structure: 1000. For $Size = 16384$, due to time limits, we reduced the number of instances to 300.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
64	$\{4\}^3$	100	1.16	0.021	0.0181	0.00033
256	$\{4\}^4$	100	1.12	0.148	0.13214	0.00058
1024	$\{4\}^5$	100	1.28	1.54	1.20312	0.0015
4096	$\{4\}^6$	100	1.26	21.532	17.0889	0.00526
16384	$\{4\}^7$	100	1.18	355.661	301.408	0.02171

Tab. 12: $Range = 10000$. Number of instances for each structure: 1000. For $Size = 3125$ and $Size = 15625$, due to time limits, we reduced the number of instances to 100.

Size	Structure	Success	Iterations	Time	Time/Iter	Time/Size
125	$\{5\}^3$	100	1.62	0.146	0.09012	0.00117
625	$\{5\}^4$	100	5.2	4.767	0.91673	0.00763
3125	$\{5\}^5$	100	7.7	122.143	15.8627	0.03909
15625	$\{5\}^6$	100	3.4	1689.16	496.812	0.10811

B Experimental Data Tables II

Tab. 13: Running times for IteratedSearch and AlternativeStrategy called on multisets with different $range$ values and $structure = \{5, 5\}$. Number of instances for each range: 100.

Size	Structure	Range	IteratedSearch	AlternativeStrategy
25	$\{5, 5\}$	100	0.09	0.144747
25	$\{5, 5\}$	300	0.008	3.764507
25	$\{5, 5\}$	500	0.009	22.003455
25	$\{5, 5\}$	700	0.008	64.317906
25	$\{5, 5\}$	900	0.007	161.541679
25	$\{5, 5\}$	1100	0.01	253.745332

Tab. 14: Running times for IteratedSearch and AlternativeStrategy called on multisets with different $range$ values and $structure = \{10, 10\}$. Number of instances for each range: 100.

Size	Structure	Range	IteratedSearch	AlternativeStrategy
100	$\{10, 10\}$	100	0.551	0.220502
100	$\{10, 10\}$	300	0.532	4.213079
100	$\{10, 10\}$	500	0.397	26.706801
100	$\{10, 10\}$	700	0.426	75.783461
100	$\{10, 10\}$	900	0.612	187.938575
100	$\{10, 10\}$	1100	0.4	379.374113

Tab. 15: Running times for IteratedSearch and AlternativeStrategy called on multisets with different $range$ values and $structure = \{2\}^{12}$. Number of instances for each range: 100.

Size	Structure	Range	IteratedSearch	AlternativeStrategy
4096	$\{2\}^{12}$	40	0.294	0.142596
4096	$\{2\}^{12}$	60	0.319	2.682864
4096	$\{2\}^{12}$	80	0.316	3.145838
4096	$\{2\}^{12}$	100	0.311	6.137466
4096	$\{2\}^{12}$	120	0.283	31.849028
4096	$\{2\}^{12}$	140	0.253	356.950613

References

I. Anderson. *Combinatorics of finite sets*. Oxford science publications. lat. Clarendon Press ; Oxford University Press, Oxford [England] : New York, 1989. ISBN 0198533799.

I. Anderson. *Combinatorics of Finite Sets*. Dover books on mathematics. Dover Publications, 2002. ISBN 9780486422572. URL https://books.google.it/books?id=RjDd4RaqrIwC.

C. Berthaud, L. Capelli, J. Gustedt, C. Kirchner, K. Loiseau, A. Magron, M. Medves, A. Monteil, G. Riverieux, and L. Romary. EPISCIENCES - an overlay publication platform. In D. P. Polydoratou, editor, *ELPUB2014 - International Conference on Electronic Publishing*, pages 78–87, Thessalonique, Greece, June 2014. Alexander Technological Education Institute of Thessaloniki, IOS Press. doi: 10.3233/978-1-61499-409-1-78. URL https://hal.inria.fr/hal-01002815.

H. Brunotte. On some classes of polynomials with nonnegative coefficients and a given factor. *Periodica Mathematica Hungarica*, 67(1):15–32, 2013. doi: 10.1007/s10998-013-2367-8. URL https://doi.org/10.1007/s10998-013-2367-8.

F. Campanini and A. Facchini. Factorizations of polynomials with integral non-negative coefficients. *Semigroup Forum*, 99(2):317–332, 2019. doi: 10.1007/s00233-018-9979-5. URL https://doi.org/10.1007/s00233-018-9979-5.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

F. Cucker, P. Koiran, and S. Smale. A polynomial time algorithm for diophantine equations in one variable. *Journal of Symbolic Computation*, 27(1):21–29, 1999. ISSN 0747-7171. doi: https://doi.org/10.1006/jsco.1998.0242. URL https://www.sciencedirect.com/science/article/pii/S0747717198902425.

M. DeVos, R. Krakovski, B. Mohar, and A. Sheikh Ahmady. Integral cayley multigraphs over abelian and hamiltonian groups. *The Electronic Journal of Combinatorics*, 20(2), Jun 2013. ISSN 1077-8926. doi: 10.37236/2742. URL http://dx.doi.org/10.37236/2742.

H. L. Dorwart. Irreducibility of polynomials. *The American Mathematical Monthly*, 42(6):369–381, 1935. doi: 10.1080/00029890.1935.11987732.

A. Dudek, A. Frieze, A. Ruciński, and M. Šileikis. Approximate counting of regular hypergraphs. *Information Processing Letters*, 113(19):785–788, 2013. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2013.07.018. URL https://www.sciencedirect.com/science/article/pii/S002001901300207X.

I. Z. Emiris, A. Karasoulou, and C. Tzovas. Approximating multidimensional subset sum and minkowski decomposition of polygons. *Mathematics in Computer Science*, 11(1):35–48, 2017. doi: 10.1007/s11786-017-0297-1. URL https://doi.org/10.1007/s11786-017-0297-1.

M. R. Fellows and N. Koblitz. Fixed-parameter complexity and cryptography. In G. D. Cohen, T. Mora, and O. Moreno, editors, *Applied algebra, algebraic algorithms and error-correcting codes, 10th International Symposium, AAECC-10, San Juan de Puerto Rico, Puerto Rico, May 10-14, 1993, Proceedings*, volume 673 of *Lecture Notes in Computer Science*, pages 121–131. Springer, 1993. ISBN 3-540-56686-4.

S. Gao and A. G. B. Lauder. Decomposition of polytopes and polynomials. *Discrete & Computational Geometry*, 26(1):89–104, 2001. doi: 10.1007/s00454-001-0024-0. URL https://doi.org/10.1007/s00454-001-0024-0.

M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979. ISBN 0716710455. URL http://www.amazon.com/Computers-Intractability-NP-Completeness-Mathematical-Sciences/dp/0716710455.

M. R. Garey and D. S.Johnson. "strong" np-completeness results: motivation, examples, and iimplications. *J. ACM*, 25(3):499–508, jul 1978. ISSN 0004-5411. doi: 10.1145/322077.322090. URL https://doi.org/10.1145/322077.322090.

B. Grenet. Bounded-degree factors of lacunary multivariate polynomials. *Journal of Symbolic Computation*, 75:171–192, 2016. ISSN 0747-7171. doi: https://doi.org/10.1016/j.jsc.2015.11.013. Special issue on the conference ISSAC 2014: Symbolic computation and computer algebra.

S. Grumbach and T. Milo. Towards tractable algebras for bags. *J. Comput. Syst. Sci.*, 52(3):570–588, 1996. doi: 10.1006/jcss.1996.0042. URL https://doi.org/10.1006/jcss.1996.0042.

F. Henglein, R. Kaarsgaard, and M. K. Mathiesen. The programming of algebra. *CoRR*, abs/2207.00850, 2022. doi: 10.48550/arXiv.2207.00850. URL https://doi.org/10.48550/arXiv.2207.00850.

M. V. Hoeij. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, 95(2):167–189, 2002. ISSN 0022-314X. doi: https://doi.org/10.1006/jnth.2001.2763. URL https://www.sciencedirect.com/science/article/pii/S0022314X01927635.

D. S. Johnson. The np-completeness column: An ongoing guide. *Journal of Algorithms*, 2(4):393–405, 1981. ISSN 0196-6774. doi: https://doi.org/10.1016/0196-6774(81)90037-7. URL https://www.sciencedirect.com/science/article/pii/0196677481900377.

E. Kaltofen. Polynomial factorization 1987-1991. In I. Simon, editor, *LATIN '92, 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992, Proceedings*, volume 583 of *Lecture Notes in Computer Science*, pages 294–313. Springer, 1992. doi: 10.1007/BFb0023837. URL https://doi.org/10.1007/BFb0023837.

E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In M. Kauers, editor, *ISSAC*, pages 208–215. ACM, 2005. ISBN 1-59593-095-7.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, 1972.

M. Karpinski and I. E. Shparlinski. On the computational hardness of testing square-freeness of sparse polynomials. In M. P. C. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *AAECC*, volume 1719 of *Lecture Notes in Computer Science*, pages 492–497. Springer, 1999. ISBN 3-540-66723-7.

K. H. Kim and F. W. Roush. Factorization of polynomials in one variable over the tropical semiring. *https://arxiv.org/abs/math/0501167*, 2005. doi: https://doi.org/10.48550/arXiv.math/0501167. URL `https://arxiv.org/abs/math/0501167`.

G. Lamperti, M. Melchiori, and M. Zanella. On multisets in database systems. volume 2235, pages 147–216, 08 2000. ISBN 978-3-540-43063-6. doi: 10.1007/3-540-45523-X_9.

A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

C. Ng, M. Barketau, T. Cheng, and M. Y. Kovalyov. Product partition and related problems of scheduling and systems reliability: computational complexity and approximation. *European Journal of Operational Research*, 207(2):601–604, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2010.05.034. URL `https://www.sciencedirect.com/science/article/pii/S0377221710003905`.

T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. *The Not So Short Introduction to LaTeX2e*, 3.3 edition, 1999. available at `http://www.loria.fr/services/tex/general/lshort2e.pdf`.

D. A. Plaisted. Sparse complex polynomials and polynomial reducibility. *J. Comput. Syst. Sci.*, 14(2): 210–221, 1977.

F. T. Schubert. De inventione divisorum. *Nova Acta Academiae Scientiarum Petropolitanae*, 11:172–182, 1793.

D. Singh, A. M. Ibrahim, T. Yohanna, and J. N. Singh. An overview of the applications of multisets. *Novi Sad Journal of Mathematics*, 37(2):73–92, 2007. URL `http://eudml.org/doc/226431`.

R. P. Stanley. *Enumerative Combinatorics*, volume 1 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2 edition, 2011. doi: 10.1017/CBO9781139058520.

R. P. Stanley and S. Fomin. *Enumerative Combinatorics*, volume 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999. doi: 10.1017/CBO9780511609589.

C. E. Van de Woestijne. Factors of disconnected graphs and polynomials with nonnegative integer coefficients. *Ars Mathematica Contemporanea*, 5(2):307–323, Apr 2012. ISSN 1855-3966. doi: 10.26493/1855-3974.202.37f. URL `http://dx.doi.org/10.26493/1855-3974.202.37f`.

A. C.-C. Yao. New algorithms in bin packing. Technical Report CS-TR-1978-662, Stanford University, Department of Computer Science, September 1978.