The basic algorithm for pseudo-Boolean programming re-re-visited

Yves Crama
HEC Management School, University of Liège, Belgium

Boolean Seminar Liblice 2023

Outline

Definitions

Pseudo-Boolean functions

A pseudo-Boolean function is a mapping $f : \{0, 1\}^n \to \mathbb{R}$, that is, a real-valued function of $0 - 1$ variables.

Definitions

Pseudo-Boolean functions

A pseudo-Boolean function is a mapping $f : \{0, 1\}^n \to \mathbb{R}$, that is, a real-valued function of $0 - 1$ variables.

Representation: tabulated form

#	x_1, x_2, x_3, x_4	$f(x_1, x_2, x_3, x_4)$
0	0,0,0,0	4
1	0,0,0,1	4
2	0,0,1,0	2
3	0,0,1,1	2
...	...	...
14	0,1,1,1	3
15	1,1,1,1	7

Multilinear representation

Multilinear polynomials

Every pseudo-Boolean function can be represented – in a unique way – as a *multilinear polynomial* in its variables.

$$f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i,$$

(Note: $x^2 = x$.)

Multilinear representation

Multilinear polynomials

Every pseudo-Boolean function can be represented – in a unique way – as a *multilinear polynomial* in its variables.

$$f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i,$$

(Note: $x^2 = x$.)

Multilinear polynomial.

$f = 4 - 9x_1 - 5x_2 - 2x_3 + 13x_1 x_2 + 13x_1 x_3 + 6x_2 x_3 x_4 - 13x_1 x_2 x_3 x_4$

Multilinear representation

Multilinear polynomials

Every pseudo-Boolean function can be represented – in a unique way – as a *multilinear polynomial* in its variables.

$$f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i,$$

(Note: $x^2 = x$.)

Multilinear polynomial.

$f = 4 - 9x_1 - 5x_2 - 2x_3 + 13x_1x_2 + 13x_1x_3 + 6x_2x_3x_4 - 13x_1x_2x_3x_4$

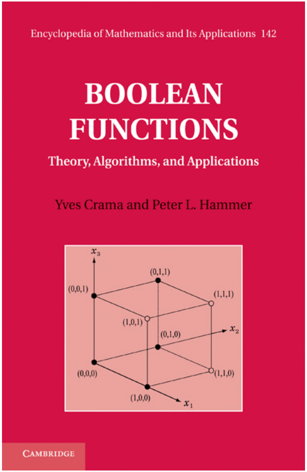
Some advertising...

Connections with Boolean functions:

BOOLEAN FUNCTIONS
Theory, Algorithms, and Applications

Yves CRAMA and Peter L. HAMMER
Cambridge University Press, 2011
710 pages

with contributions by C. Benzaken, E. Boros, N. Brauner, M.C. Golumbic, V. Gurvich, L. Hellerstein, T. Ibaraki, A. Kogan, K. Makino, B. Simeone



Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x \in \{0,1\}^n} f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x \in \{0,1\}^n} f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Complexity

PUB is NP-hard if f is a multilinear polynomial of degree 2 or more.

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x \in \{0,1\}^n} f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Complexity

PUB is NP-hard if f is a multilinear polynomial of degree 2 or more.

Numerous applications in various fields, e.g,

- naturally models satisfiability and maximum satisfiability (in particular, MAX 2SAT)
- MAX CUT, MAX STABLE SET
- implementation of quantum computing

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x \in \{0,1\}^n} f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x \in \{0,1\}^n} f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Numerous applications in various fields, e.g,

- Telecommunications and statistical mechanics.
- Compute $\{-1, +1\}$ sequences $s = (s_1, \ldots, s_n)$ with low auto-correlations.

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x\in\{0,1\}^n} f(x) = \sum_{S\in\mathcal{M}} a_S \prod_{k\in S} x_k + \sum_{i=1}^{n} a_i x_i$$

Numerous applications in various fields, e.g,

- Telecommunications and statistical mechanics.
- Compute $\{-1,+1\}$ sequences $s = (s_1, \ldots, s_n)$ with low auto-correlations.
- Given $r \leq n$, find $s \in \{-1,+1\}^n$ to minimize

$$E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2 .$$

- $E_{n,r}$ is a polynomial of degree 4 (easily transformed to 0-1 variables).
- Very hard for MIP solvers as soon as $n, r \geq 40$ (instances on MINLPLib).

Polynomial unconstrained optimization in binary variables

$$\text{(PUB)} \quad \min_{x\in\{0,1\}^n} f(x) = \sum_{S\in\mathcal{M}} a_S \prod_{k\in S} x_k + \sum_{i=1}^{n} a_i x_i$$

- Some classical approaches:
  - *Linearization*.
  - *Quadratization*.
  - *Variable elimination*.

Outline

Basic algorithm

- A dynamic programming algorithm based on variable elimination (Hammer, Rosenberg and Rudeanu 1963)

Hammer, P.L., Rosenberg, I., Rudeanu, S., 1963. On the determination of the minima of pseudo- Boolean functions. *Studii si Cercetari Matematice* 14, 359-364. In Romanian.

Hammer, P.L., Rudeanu, S., 1968. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, Berlin.

Basic algorithm

- A dynamic programming algorithm based on variable elimination (Hammer, Rosenberg and Rudeanu 1963)
- Re-visited by Crama, Hansen and Jaumard (1990)

Crama, Y., Hansen, P., Jaumard, B., 1990. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Applied Mathematics* 29, 171-185.

Basic algorithm

- A dynamic programming algorithm based on variable elimination (Hammer, Rosenberg and Rudeanu 1963)
- Re-visited by Crama, Hansen and Jaumard (1990)
- Re-re-visited in 2023!

Clausen, J.V., Crama, Y., Lusby, R., Rodriguez-Heck, E. and Ropke, S. 2023. Solving unconstrained binary polynomial programs with limited reach. Working paper, HEC-ULiege.

Basic algorithm

- Central idea: inspired by classical elimination methods for Boolean equations (Boole 1854).
- The Boolean equation $\varphi(x_1, x_2, \ldots, x_n) = 0$ is consistent if and only if the equation

$$\phi(x_2, \ldots, x_n) = \varphi(0, x_2, \ldots, x_n)\,\varphi(1, x_2, \ldots, x_n) = 0 \tag{1}$$

is consistent.

Basic algorithm

- Central idea: inspired by classical elimination methods for Boolean equations (Boole 1854).
- The Boolean equation $\varphi(x_1, x_2, \dots, x_n) = 0$ is consistent if and only if the equation

$$\phi(x_2, \dots, x_n) = \varphi(0, x_2, \dots, x_n)\,\varphi(1, x_2, \dots, x_n) = 0 \qquad (1)$$

  is consistent.
- Repeat until all variables are eliminated.

Basic algorithm

- Central idea: inspired by classical elimination methods for Boolean equations (Boole 1854).
- The Boolean equation $\varphi(x_1, x_2, \ldots, x_n) = 0$ is consistent if and only if the equation

$$\phi(x_2, \ldots, x_n) = \varphi(0, x_2, \ldots, x_n)\,\varphi(1, x_2, \ldots, x_n) = 0 \qquad (1)$$

  is consistent.
- Repeat until all variables are eliminated.
- Note:

$$\phi(x_2, \ldots, x_n) = \min(\varphi(0, x_2, \ldots, x_n), \varphi(1, x_2, \ldots, x_n)).$$

Basic algorithm

For pseudo-Boolean optimization:

- Let $f_1(x_1, \ldots, x_n) := f(x_1, \ldots, x_n)$.
- Eliminate x_1, that is, produce an expression of the function

$$f_2(x_2, \ldots, x_n) \triangleq \min_{x_1} f_1(x_1, \ldots, x_n) = \min(f_1(0, x_2, \ldots, x_n), f_1(1, x_2, \ldots, x_n)).$$

Basic algorithm

For pseudo-Boolean optimization:

- Let $f_1(x_1, \ldots, x_n) := f(x_1, \ldots, x_n)$.
- Eliminate x_1, that is, produce an expression of the function

$$f_2(x_2, \ldots, x_n) \triangleq \min_{x_1} f_1(x_1, \ldots, x_n) = \min(f_1(0, x_2, \ldots, x_n), f_1(1, x_2, \ldots, x_n)).$$

- How?

Basic algorithm

For pseudo-Boolean optimization:

- Let $f_1(x_1, \dots, x_n) := f(x_1, \dots, x_n)$.
- Eliminate x_1, that is, produce an expression of the function

$$f_2(x_2, \dots, x_n) \triangleq \min_{x_1} f_1(x_1, \dots, x_n) = \min(f_1(0, x_2, \dots, x_n), f_1(1, x_2, \dots, x_n)).$$

- How? Write $f_1(x_1, \dots, x_n) = x_1 g(x_2, \dots, x_n) + h(x_2, \dots, x_n)$.
  (Straighforward if f is in polynomial form.)

Basic algorithm

For pseudo-Boolean optimization:

- Let $f_1(x_1, \ldots, x_n) := f(x_1, \ldots, x_n)$.
- Eliminate x_1, that is, produce an expression of the function

$$f_2(x_2, \ldots, x_n) \triangleq \min_{x_1} f_1(x_1, \ldots, x_n) = \min(f_1(0, x_2, \ldots, x_n), f_1(1, x_2, \ldots, x_n)).$$

- How? Write $f_1(x_1, \ldots, x_n) = x_1 g(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$. (Straighforward if f is in polynomial form.)
- For any $(x_2, \ldots, x_n)$,
  - $f_1(0, x_2, \ldots, x_n) = h(x_2, \ldots, x_n)$
  - $f_1(1, x_2, \ldots, x_n) = g(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$
- So: $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.

Basic algorithm

For pseudo-Boolean optimization:

- Let $f_1(x_1, \ldots, x_n) := f(x_1, \ldots, x_n)$.
- Eliminate x_1, that is, produce an expression of the function

$$f_2(x_2, \ldots, x_n) \triangleq \min_{x_1} f_1(x_1, \ldots, x_n) = \min(f_1(0, x_2, \ldots, x_n), f_1(1, x_2, \ldots, x_n)).$$

- How? Write $f_1(x_1, \ldots, x_n) = x_1 g(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$. (Straighforward if f is in polynomial form.)
- For any $(x_2, \ldots, x_n)$,
  - $f_1(0, x_2, \ldots, x_n) = h(x_2, \ldots, x_n)$
  - $f_1(1, x_2, \ldots, x_n) = g(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$
- So: $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Repeat until all variables are eliminated.

Basic algorithm

Crucial step:

- $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \ldots, x_n)\}$.

Basic algorithm

Crucial step:

- $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \ldots, x_n)\}$.
- Previous attempts: obtain a polynomial expression of ψ.

Basic algorithm

Crucial step:

- $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \ldots, x_n)\}$.
- Previous attempts: obtain a polynomial expression of ψ.
- Hammer, Rosenberg and Rudeanu (1963) proceed by algebraic manipulations - never implemented.

Basic algorithm

Crucial step:

- $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \ldots, x_n)\}$.
- Previous attempts: obtain a polynomial expression of ψ.
- Hammer, Rosenberg and Rudeanu (1963) proceed by algebraic manipulations - never implemented.
- Crama, Hansen and Jaumard (1990) propose a branch-and-bound algorithm to compute ψ.

Basic algorithm

Crucial step:

- $f_2(x_2, \dots, x_n) = \min\{0, g(x_2, \dots, x_n)\} + h(x_2, \dots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \dots, x_n)\}$.
- Previous attempts: obtain a polynomial expression of ψ.
- Hammer, Rosenberg and Rudeanu (1963) proceed by algebraic manipulations - never implemented.
- Crama, Hansen and Jaumard (1990) propose a branch-and-bound algorithm to compute ψ.
- Observe: if $g(x_2, \dots, x_n)$ depends on a bounded number of variables (say, w variables), then an expression of ψ can be computed in time $O(2^w)$:

$$\psi = \sum_{S \subseteq [w]} a_S \prod_{i \in S} x_i.$$

Basic algorithm

Crucial step:

- $f_2(x_2, \ldots, x_n) = \min\{0, g(x_2, \ldots, x_n)\} + h(x_2, \ldots, x_n)$.
- Compute $\psi = \min\{0, g(x_2, \ldots, x_n)\}$.
- Previous attempts: obtain a polynomial expression of ψ.
- Hammer, Rosenberg and Rudeanu (1963) proceed by algebraic manipulations - never implemented.
- Crama, Hansen and Jaumard (1990) propose a branch-and-bound algorithm to compute ψ.
- Observe: if $g(x_2, \ldots, x_n)$ depends on a bounded number of variables (say, w variables), then an expression of ψ can be computed in time $O(2^w)$:

$$\psi = \sum_{S \subseteq [w]} a_S \prod_{i \in S} x_i.$$

- This happens at all iterations of the basic algorithm if the co-occurrence graph of f has *treewidth* at most w.

Co-occurrence graph

Co-occurrence graph of a function $f(x) = \sum_{S \in \mathcal{M}} a_S \prod_{k \in S} x_k + \sum_{i=1}^{n} a_i x_i$:

- vertices = variables
- $\{x_i, x_j\}$ is an edge if x_i and x_j appear in a same monomial S.

Example:

$f = 4 - 9x_1 - 5x_2 - 2x_3 + 13x_1x_2 + 13x_1x_3 + 6x_2x_3x_4$.
Edges: $\{x_1, x_2\}$, $\{x_1, x_3\}$,$\{x_2, x_3\}$, $\{x_2, x_4\}$, $\{x_3, x_4\}$.

Basic algorithm with bounded treewidth

Treewidth w

There is an elimination ordering $x_1, \ldots, x_n$ of vertices such that, if we perform the following operations for $i = 1, \ldots, n$,

- replace the neighborhood of vertex x_i by a clique and remove x_i from the graph,

then x_i has at most w neighbors at each iteration.

Basic algorithm with bounded treewidth

Treewidth w

There is an elimination ordering $x_1, \ldots, x_n$ of vertices such that, if we perform the following operations for $i = 1, \ldots, n$,

- replace the neighborhood of vertex x_i by a clique and remove x_i from the graph,

then x_i has at most w neighbors at each iteration.

Crama, Hansen and Jaumard (1990)

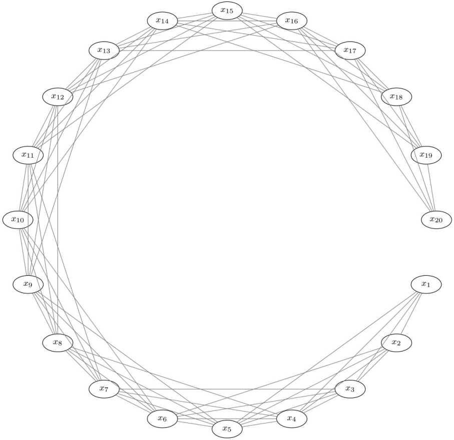
When the co-occurrence graph of f has bounded treewidth w, the basic algorithm can be implemented to run in time $O(n2^w)$.

Low-autocorrelation sequence instances

- Co-occurrence graph for low-autocorrelation sequence instances:
  $\min E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2$
  with $n = 20, r = 5$.

Low-autocorrelation sequence instances

- Co-occurrence graph for low-autocorrelation sequence instances:
  $\min E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2$
  with $n = 20, r = 5$.



Basic algorithm with bounded treewidth

- Low-autocorrelation sequence instance $E_{n,r}$ has treewidth at most r.

Basic algorithm with bounded treewidth

- Low-autocorrelation sequence instance $E_{n,r}$ has treewidth at most r.
- Even better, $E_{n,r}$ has bounded *reach* r: if variables x_i and x_j appear together in a monomial, then $|i - j| \leq r$.

Basic algorithm with bounded treewidth

- Low-autocorrelation sequence instance $E_{n,r}$ has treewidth at most r.
- Even better, $E_{n,r}$ has bounded *reach* r: if variables x_i and x_j appear together in a monomial, then $|i - j| \leq r$.
- If x_i has lowest index in a term $\prod_{k \in S} x_k$, then $S \subseteq \{i, i+1, \ldots, i+r-1\}$.

Basic algorithm with bounded treewidth

- Low-autocorrelation sequence instance $E_{n,r}$ has treewidth at most r.
- Even better, $E_{n,r}$ has bounded *reach* r: if variables x_i and x_j appear together in a monomial, then $|i-j| \leq r$.
- If x_i has lowest index in a term $\prod_{k \in S} x_k$, then $S \subseteq \{i, i+1, \ldots, i+r-1\}$.
- Note: reach $r \Rightarrow$ treewidth $\leq r-1$.

Basic algorithm with bounded treewidth

- Low-autocorrelation sequence instance $E_{n,r}$ has treewidth at most r.
- Even better, $E_{n,r}$ has bounded *reach* r: if variables x_i and x_j appear together in a monomial, then $|i - j| \leq r$.
- If x_i has lowest index in a term $\prod_{k \in S} x_k$, then $S \subseteq \{i, i+1, \ldots, i+r-1\}$.
- Note: reach $r \Rightarrow$ treewidth $\leq r - 1$.
- Allows a different implementation of the basic algorithm (Clausen et al. 2023).

Basic algorithm with bounded reach

- After elimination of $x_1, \ldots, x_{t-1}$, let

$$f_t(x_t, \ldots, x_n) = \min_{x_1, \ldots, x_{t-1}} f_1(x_1, \ldots, x_n).$$

Basic algorithm with bounded reach

- After elimination of $x_1, \ldots, x_{t-1}$, let

$$f_t(x_t, \ldots, x_n) = \min_{x_1, \ldots, x_{t-1}} f_1(x_1, \ldots, x_n).$$

- Maintain f_t as

$$f_t(x_t, \ldots, x_n) = C_t(x_t, \ldots, x_{t+r-1}) + L_t(x_{t+1}, \ldots, x_n)$$

where

$$L_t(x_{t+1}, \ldots, x_n) = sc_{t+1}(x_{t+1}, \ldots, x_{t+r}) + L_{t+1}(x_{t+2}, \ldots, x_n),$$

L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.

Basic algorithm with bounded reach

- After elimination of $x_1, \ldots, x_{t-1}$, let

$$f_t(x_t, \ldots, x_n) = \min_{x_1, \ldots, x_{t-1}} f_1(x_1, \ldots, x_n).$$

- Maintain f_t as

$$f_t(x_t, \ldots, x_n) = C_t(x_t, \ldots, x_{t+r-1}) + L_t(x_{t+1}, \ldots, x_n)$$

where

$$L_t(x_{t+1}, \ldots, x_n) = sc_{t+1}(x_{t+1}, \ldots, x_{t+r}) + L_{t+1}(x_{t+2}, \ldots, x_n),$$

L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.

- Initially,

$$f_1(x_1, \ldots, x_n) = x_1 g(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$$

with $C_1(x_1, \ldots, x_r) = x_1 g(x_2, \ldots, x_n)$.

Basic algorithm with bounded reach

- Maintain f_t as

$$f_t(x_t, \ldots, x_n) = C_t(x_t, \ldots, x_{t+r-1}) + sc_{t+1}(x_{t+1}, \ldots, x_{t+r}) + L_{t+1}(x_{t+2}, \ldots, x_n),$$

where L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.

Basic algorithm with bounded reach

- Maintain f_t as

$$f_t(x_t,\ldots,x_n) = C_t(x_t,\ldots,x_{t+r-1}) + sc_{t+1}(x_{t+1},\ldots,x_{t+r}) + L_{t+1}(x_{t+2},\ldots,x_n),$$

where L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.

- Then

$$\begin{aligned} & f_{t+1}(x_{t+1},\ldots,x_n) \\ &= \min_{x_t} \{C_t(x_t,\ldots,x_{t+r-1}) + sc_{t+1}(x_{t+1},\ldots,x_{t+r})\} + L_{t+1}(x_{t+2},\ldots,x_n) \\ &= C_{t+1}(x_{t+1},\ldots,x_{t+r}) + L_{t+1}(x_{t+2},\ldots,x_n). \end{aligned}$$

Basic algorithm with bounded reach

- Maintain f_t as

  $$f_t(x_t, \dots, x_n) = C_t(x_t, \dots, x_{t+r-1}) + sc_{t+1}(x_{t+1}, \dots, x_{t+r}) + L_{t+1}(x_{t+2}, \dots, x_n),$$

  where L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.
- Then

  $$\begin{aligned} & f_{t+1}(x_{t+1}, \dots, x_n) \\ & = \min_{x_t} \{C_t(x_t, \dots, x_{t+r-1}) + sc_{t+1}(x_{t+1}, \dots, x_{t+r})\} + L_{t+1}(x_{t+2}, \dots, x_n) \\ & = C_{t+1}(x_{t+1}, \dots, x_{t+r}) + L_{t+1}(x_{t+2}, \dots, x_n). \end{aligned}$$

- If the values of C_t are tabulated for all $(x_t, \dots, x_{t+r-1})$, then the values of C_{t+1} can be easily tabulated for all $(x_{t+1}, \dots, x_{t+r})$.

Basic algorithm with bounded reach

- Maintain f_t as

  $$f_t(x_t, \ldots, x_n) = C_t(x_t, \ldots, x_{t+r-1}) + sc_{t+1}(x_{t+1}, \ldots, x_{t+r}) + L_{t+1}(x_{t+2}, \ldots, x_n),$$

  where L_t consists of terms of f and sc_{t+1} contains the terms of f having x_{t+1} as first variable.
- Then

  $$\begin{aligned} & f_{t+1}(x_{t+1}, \ldots, x_n) \\ & = \min_{x_t} \{C_t(x_t, \ldots, x_{t+r-1}) + sc_{t+1}(x_{t+1}, \ldots, x_{t+r})\} + L_{t+1}(x_{t+2}, \ldots, x_n) \\ & = C_{t+1}(x_{t+1}, \ldots, x_{t+r}) + L_{t+1}(x_{t+2}, \ldots, x_n). \end{aligned}$$

- If the values of C_t are tabulated for all $(x_t, \ldots, x_{t+r-1})$, then the values of C_{t+1} can be easily tabulated for all $(x_{t+1}, \ldots, x_{t+r})$.
- Each table has size 2^r.

Properties of New BA

- This New BA is polynomial $O(n2^r)$ for instances with reach r (Clausen et al. 2023).

Properties of New BA

- This New BA is polynomial $O(n2^r)$ for instances with reach r (Clausen et al. 2023).
- New BA avoids the computation of polynomial expressions.

Properties of New BA

- This New BA is polynomial $O(n2^r)$ for instances with reach r (Clausen et al. 2023).
- New BA avoids the computation of polynomial expressions.
- Each iteration step of New BA can be parallelized.

Computational results for New BA

Given $r \leq n$, find s to minimize

$$E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2 .$$

Computational results for New BA

Given $r \leq n$, find s to minimize

$$E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2 .$$

Results with New BA:

- For the low-autocorrelation binary sequence problem, the New BA performs much better than linearization, quadratization, or previous versions of the basic algorithm (Old BA).

Computational results for New BA

Given $r \leq n$, find s to minimize

$$E_{n,r}(s) = \sum_{i=1}^{n-r+1} \sum_{d=1}^{r-1} \left(\sum_{j=i}^{i+r-1-d} s_j s_{j+d} \right)^2 .$$

Results with New BA:

- For the low-autocorrelation binary sequence problem, the New BA performs much better than linearization, quadratization, or previous versions of the basic algorithm (Old BA).
- 10 instances are solved to optimality for the first time. For example:
  - Instance 40.20: cannot be solved in 3 hours by linearization (gap > 100%) nor by PQCR (gap = 4%); solved in 450 sec by Old BA, in 9 sec by New BA.
  - Instance 50.25: cannot be solved in 3 hours by linearization (gap > 100%) nor by PQCR (gap = 11%) nor by Old BA (runs out of memory); solved in 468 sec by New BA.
  - BA struggles when r gets large; largest instances solved: 55.28 and 60.15.

Outline

Conclusions

- Polynomial unconstrained binary optimization problems are very hard nuts!

Conclusions

- Polynomial unconstrained binary optimization problems are very hard nuts!
- Old ideas are still fruitful: linearization (1959), quadratization (1975), variable elimination (1963).

Conclusions

- Polynomial unconstrained binary optimization problems are very hard nuts!
- Old ideas are still fruitful: linearization (1959), quadratization (1975), variable elimination (1963).
- Algorithms must be tailored carefully, must often be specifically adapted for the problem at hand.

Conclusions

- Polynomial unconstrained binary optimization problems are very hard nuts!
- Old ideas are still fruitful: linearization (1959), quadratization (1975), variable elimination (1963).
- Algorithms must be tailored carefully, must often be specifically adapted for the problem at hand.
- Still a lot of work to do, both theoretical and computational.

Bibliography

- Hammer, P.L., Rosenberg, I., Rudeanu, S., 1963b. On the determination of the minima of pseudo- Boolean functions. *Studii si Cercetari Matematice* 14, 359-364. In Romanian.
- Hammer, P.L., Rudeanu, S., 1968. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, Berlin.
- Crama, Y., Hansen, P., Jaumard, B., 1990. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Applied Mathematics* 29, 171-185.
- Clausen, J.V., Crama, Y., Lusby, R., Rodriguez-Heck, E. and Ropke, S. Solving unconstrained binary polynomial programs with limited reach. Working paper, HEC-ULiege, 2023.