Quadratic Binary Optimization and Its Applications

Implication Networks and Persistencies

Endre Boros

RUTCOR, Rutgers University

Fields Institute, Toronto, October 7, 2008.

Joint work with P.L. Hammer[1] and G. Tavares

[1] (1936-2006)

Outline

Quadratic Unconstrained Binary Optimization (QUBO)

Variables and Literals

- **Variables**: $x_1,\ x_2,\ \ldots,\ x_n \quad \in \{0,1\}$.
- **Negations**: $\overline{x}_i \ = \ 1 - x_i \quad \in \{0,1\}$ for $i = 1, \ldots, n$

Quadratic Pseudo-Boolean Function (QPBF): $f : \{0,1\}^n \to \mathbb{R}$

$$f(x_1, \ldots, x_n) \ = \ c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \leq i < j \leq n} c_{ij} x_i x_j$$

Quadratic Unconstrained Binary Optimization (QUBO)

$$\min_{(x_1,\ldots,x_n)\in\{0,1\}^n} f(x_1, \ldots, x_n)$$

(Hammer and Rudeanu, 1968)

Quadratic Unconstrained Binary Optimization (QUBO)

Variables and Literals

- **Variables**: $x_1, x_2, \ldots, x_n \quad \in \{0,1\}$.
- **Negations**: $\overline{x}_i \; = \; 1 - x_i \quad \in \{0,1\}$ for $i = 1, \ldots, n$

Quadratic Pseudo-Boolean Function (QPBF): $f : \{0,1\}^n \to \mathbb{R}$

$$f(x_1, \ldots, x_n) \; = \; c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \le i < j \le n} c_{ij} x_i x_j$$

Quadratic Unconstrained Binary Optimization (QUBO)

$$\min_{(x_1,\ldots,x_n)\in\{0,1\}^n} f(x_1, \ldots, x_n)$$

Quadratic Unconstrained Binary Optimization (QUBO)

Variables and Literals

- **Variables**: $x_1, x_2, \ldots, x_n \quad \in \{0,1\}$.
- **Negations**: $\overline{x}_i \;=\; 1 - x_i \quad \in \{0,1\}$ for $i = 1, \ldots, n$

Quadratic Pseudo-Boolean Function (QPBF): $f : \{0,1\}^n \to \mathbb{R}$

$$f(x_1, \ldots, x_n) \;=\; c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \le i < j \le n} c_{ij} x_i x_j$$

Quadratic Unconstrained Binary Optimization (QUBO)

$$\min_{(x_1,\ldots,x_n)\in\{0,1\}^n} f(x_1, \ldots, x_n)$$

Pseudo-Boolean Optimization

(Hammer and Rudeanu, 1968)

Quadratic Unconstrained Binary Optimization (QUBO)

Variables and Literals

- **Variables**: $x_1,\ x_2,\ \dots,\ x_n \quad \in \{0,1\}.$
- **Negations**: $\overline{x}_i \ = \ 1 - x_i \quad \in \{0,1\}$ for $i = 1, \dots, n$

Quadratic Pseudo-Boolean Function (QPBF): $f : \{0,1\}^n \longrightarrow \mathbb{R}$

$$f(x_1, \dots, x_n) \ = \ c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \le i < j \le n} c_{ij} x_i x_j$$

Quadratic Unconstrained Binary Optimization (QUBO)

$$\min_{(x_1,\dots,x_n)\in\{0,1\}^n} f(x_1, \dots, x_n)$$

Pseudo-Boolean Optimization

(Hammer and Rudeanu, 1968)

Quadratic Unconstrained Binary Optimization (QUBO)

Variables and Literals

- **Variables**: $x_1, x_2, ..., x_n \quad \in \{0, 1\}$.
- **Negations**: $\overline{x}_i \; = \; 1 - x_i \quad \in \{0, 1\}$ for $i = 1, ..., n$

Quadratic Pseudo-Boolean Function (QPBF): $f : \{0, 1\}^n \longrightarrow \mathbb{R}$

$$f(x_1, ..., x_n) \; = \; c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \le i < j \le n} c_{ij} x_i x_j$$

Quadratic Unconstrained Binary Optimization (QUBO)

$$\min_{(x_1, ..., x_n) \in \{0,1\}^n} f(x_1, ..., x_n)$$

Pseudo-Boolean Optimization

(Hammer and Rudeanu, 1968)

Outline

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Applications of QUBO

- MAX-2-SAT, MAXCUT, Maximum Stable Set, Maximum Clique, Graph Balancing, ...
  - Physics (Ising problem)
  - VLSI Design (via minimization, floor partitioning, wire length minimization, verification, buffer assignment, ...)
  - Finance (capital budgeting, portfolio optimization)
  - Image Processing (segmentation, denoising, deblurring, MRI, ...)
  - Statistics (clustering, maximum likelihood ranking)
  - Manufacturing (scheduling, production, location, ...)

Outline

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$f = -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 \qquad \text{QPBF}$$

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$
\begin{aligned}
f &= -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{QPBF} \\
&= -5 + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform}
\end{aligned}
$$

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$
\begin{aligned}
f &= -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{QPBF} \\
&= \mathbf{-5} + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform} \\
&= \mathbf{-4} + \overline{x}_3 + \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform}
\end{aligned}
$$

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$\begin{aligned}
f &= -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{QPBF} \\
&= -5 + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform} \\
&= -4 + \overline{x}_3 + \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform} \\
&= -3 + x_1x_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3 && \text{cubic posiform}
\end{aligned}$$

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$\begin{aligned} f &= -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{QPBF} \\ &= \mathbf{-5} + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform} \\ &= \mathbf{-4} + \overline{x}_3 + \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_3 && \text{quadratic posiform} \\ &= \mathbf{-3} + x_1x_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3 && \text{cubic posiform} \end{aligned}$$

MAX2SAT $\equiv$ minimization of a quadratic posiform $\equiv$ **QUBO**.

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

$$
\begin{array}{rcll}
f & = & -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 & \text{QPBF} \\
 & = & \mathbf{-5} + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 & \text{quadratic posiform} \\
 & = & \mathbf{-4} + \overline{x}_3 + \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_3 & \text{quadratic posiform} \\
 & = & \mathbf{-3} + x_1x_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3 & \text{cubic posiform}
\end{array}
$$

MAX2SAT $\equiv$ minimization of a quadratic posiform $\equiv$ **QUBO**.

Roof Dual Bound: $C_2(f) \leq f$ (Hammer, Hansen and Simeone, 1984)

$\mathbf{C_2}(f) =$ largest C s.t. $f = C + \phi$ for some **quadratic posiform** ϕ.

Representations and Lower Bounds

Posiforms: Nonnegative (except maybe the constant terms) multi-linear polynomials in $2n$ literals x_1, $\overline{x}_1$, ..., x_n, $\overline{x}_n$

f	$=$	$-2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3$	QPBF
	$=$	$\mathbf{-5} + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3$	quadratic posiform
	$=$	$\mathbf{-4} + \overline{x}_3 + \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_3$	quadratic posiform
	$=$	$\mathbf{-3} + x_1x_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3$	cubic posiform

MAX2SAT $\equiv$ minimization of a quadratic posiform $\equiv$ **QUBO**.

Roof Dual Bound: $C_2(f) \leq f$ (Hammer, Hansen and Simeone, 1984)

$\mathbf{C_2}(f)$ = largest C s.t. $f = C + \phi$ for some **quadratic posiform** ϕ.

Complete Hierarchy of Bounds: (B, Crama and Hammer, 1990)

$$\mathbf{C_2}(f) \leq \mathbf{C_3}(f) \leq \cdots \leq \mathbf{C_n}(f) = \min f$$

Outline

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

y is a **persistency** for a pseudo-Boolean function f if

$$f(x|y) \leq f(x) \quad \forall \;\; x \in \{0,1\}^V.$$

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

y is a **persistency** for a pseudo-Boolean function f if

$$f(x|y) \leq f(x) \quad \forall \quad x \in \{0,1\}^V.$$

y is an **autarky** for a posiform ϕ if

$T(y) = 0$ for all terms T of ϕ for which $Var(T) \cap S \neq \emptyset$.

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

y is a **persistency** for a pseudo-Boolean function f if

$$f(x|y) \leq f(x) \quad \forall \quad x \in \{0,1\}^V.$$

y is an **autarky** for a posiform ϕ if

$T(y) = 0$ for all terms T of ϕ for which $Var(T) \cap S \neq \emptyset$.

$y = (\mathbf{1}, \mathbf{1}, \mathbf{1}, *, *, *, *)$ is an autarky of the posiform

$$\phi = \mathbf{x_1 \overline{x}_2} + 5\mathbf{\overline{x}_1 x_3} x_6 + 4\mathbf{x_2 \overline{x}_3} x_7 + 4\mathbf{\overline{x}_1} x_4 + 5\mathbf{\overline{x}_2} x_5 + 6x_4 x_5$$

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

y is a persistency for a pseudo-Boolean function f if

$$f(x|y) \leq f(x) \quad \forall \quad x \in \{0,1\}^V.$$

y is an autarky for a posiform ϕ if

$T(y) = 0$ for all terms T of ϕ for which $Var(T) \cap S \neq \emptyset$.

Given y and $f = \phi$, it is

- easy to test if y is an autarky for ϕ;
- hard to test if y is a persistency for f.

Persistencies and Autarkies

Partial assignment: $y \in \{0,1\}^S$, $S \subseteq V$

y is a **persistency** for a pseudo-Boolean function f if

$$f(x|y) \leq f(x) \quad \forall \quad x \in \{0,1\}^V.$$

y is an **autarky** for a posiform ϕ if

$T(y) = 0$ for all terms T of ϕ for which $Var(T) \cap S \neq \emptyset$.

Given y and $f = \phi$, it is

- **easy** to test if y is an autarky for ϕ;
- **hard** to test if y is a persistency for f.

Basic facts about persistencies and autarkies

An autarky y of a posiform ϕ is a persistency of the function f represented by ϕ.

Basic facts about persistencies and autarkies

An autarky y of a posiform ϕ is a persistency of the function f represented by ϕ.

A persistency y of the function f is an autarky for some posiform ϕ representing f.

Basic facts about persistencies and autarkies

An autarky y of a posiform ϕ is a persistency of the function f represented by ϕ.

A persistency y of the function f is an autarky for some posiform ϕ representing f.

If f has persistencies $y^1 \in \{0,1\}^{S_1}$ and $y^2 \in \{0,1\}^{S_2}$, then it also has a persistency $y^3 \in \{0,1\}^{S_1 \cup S_2}$.

Basic facts about persistencies and autarkies

An autarky y of a posiform ϕ is a persistency of the function f represented by ϕ.

A persistency y of the function f is an autarky for some posiform ϕ representing f.

If f has persistencies $y^1 \in \{0,1\}^{S_1}$ and $y^2 \in \{0,1\}^{S_2}$, then it also has a persistency $y^3 \in \{0,1\}^{S_1 \cup S_2}$.

A posiform ϕ has a unique maximal subset $S = S(\phi)$ for which it has an autarky $y \in \{0,1\}^S$. For a **quadratic** posiform ϕ it is **easy to find $S(\phi)$**. (B. and Hammer, 1990)

Outline

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x \in \{0,1\}^n} f(x) = \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
  - fix persistent variables in set $S(\phi_k)$ and simplify the problem;
  - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent
- **How to find $S(\phi)$? How to manipulate posiforms?**

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x\in\{0,1\}^n} f(x) \;=\; \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
  - fix persistent variables in set $S(\phi_k)$, and simplify the problem;
  - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent.
- **How to find $S(\phi)$? How to manipulate posiforms?**

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x \in \{0,1\}^n} f(x) \;=\; \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
    - fix persistent variables in set $S(\phi_k)$, and simplify the problem;
    - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent.
- How to find $S(\phi)$? How to manipulate posiforms?

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x\in\{0,1\}^n} f(x) \;=\; \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
  - fix persistent variables in set $S(\phi_k)$, and simplify the problem;
  - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent.
- **How to find $S(\phi)$? How to manipulate posiforms?**

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x \in \{0,1\}^n} f(x) = \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
  - fix persistent variables in set $S(\phi_k)$, and simplify the problem;
  - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent.
- How to find $S(\phi)$? How to manipulate posiforms?

Use of posiforms for QUBO

- Posiforms provide autarkies (persistencies) $S(\phi)$
- Denoting by $C(\phi)$ the constant term of a posiform ϕ, we have

$$\min_{x \in \{0,1\}^n} f(x) \;=\; \max\{C(\phi) \mid \phi \text{ is a posiform of } f\}$$

- QUBO can be solved by finding better and better posiform representations of the objective; for each posiform ϕ_k
  - fix persistent variables in set $S(\phi_k)$, and simplify the problem;
  - try to generate from ϕ_k another posiform ϕ_{k+1} such that $C(\phi_k) < C(\phi_{k+1})$, until all variables become persistent.
- **How to find $S(\phi)$? How to manipulate posiforms?**

Outline

1. Quadratic Unconstrained Binary Optimization
   - Quadratic Pseudo-Boolean Functions
   - Applications of QUBO
   - Representations and Bounds
   - Persistencies and Autarkies
   - Posiforms and QUBO
   - Implication Networks
   - Graph Cuts and Implication Networks
2. Results
   - Components of the Algorithm
   - Computational Results
   - References

QPBF $\longrightarrow$ Posiform

$$f = \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3}$$

QPBF $\longrightarrow$ Posiform

$$
\begin{aligned}
f &= \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3} \\
&= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1(1 - \overline{x}_3) + 4x_2x_3
\end{aligned}
$$

QPBF $\longrightarrow$ Posiform

$$
\begin{aligned}
f &= \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3} \\
&= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1(1 - \overline{x}_3) + 4x_2x_3 \\
&= 10 - 4x_1 - 6x_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3
\end{aligned}
$$

QPBF $\longrightarrow$ Posiform

$$
\begin{aligned}
f &= \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3} \\
&= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1(1 - \overline{x}_3) + 4x_2x_3 \\
&= 10 - 4x_1 - 6x_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 10 - 4(1 - \overline{x}_1) - 6(1 - \overline{x}_2) + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3
\end{aligned}
$$

QPBF $\longrightarrow$ Posiform

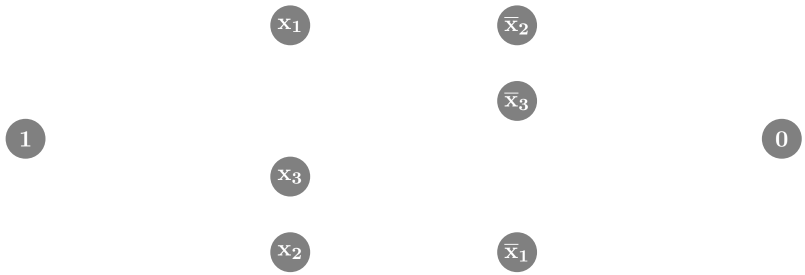
$$
\begin{aligned}
f &= \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3} \\
&= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1(1 - \overline{x}_3) + 4x_2x_3 \\
&= 10 - 4x_1 - 6x_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 10 - 4(1 - \overline{x}_1) - 6(1 - \overline{x}_2) + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= \mathbf{4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3} \;=\; \phi
\end{aligned}
$$

QPBF $\longrightarrow$ Posiform

$$
\begin{aligned}
f &= \mathbf{10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3} \\
&= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1(1 - \overline{x}_3) + 4x_2x_3 \\
&= 10 - 4x_1 - 6x_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 10 - 4(1 - \overline{x}_1) - 6(1 - \overline{x}_2) + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= \mathbf{4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3} \;=\; \phi
\end{aligned}
$$

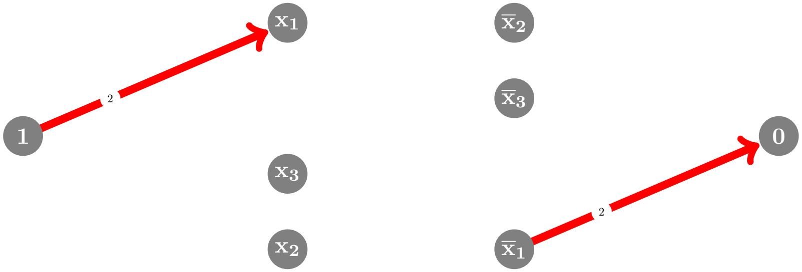
$$\mathbf{C}(\phi) = \mathbf{0} \quad \text{and} \quad \mathbf{S}(\phi) = \emptyset$$

Implication Networks



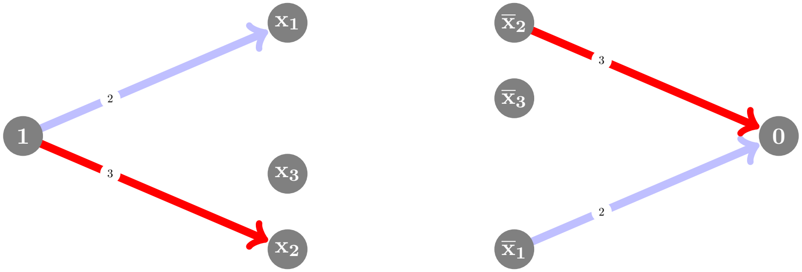
$$f = 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3$$

Implication Networks



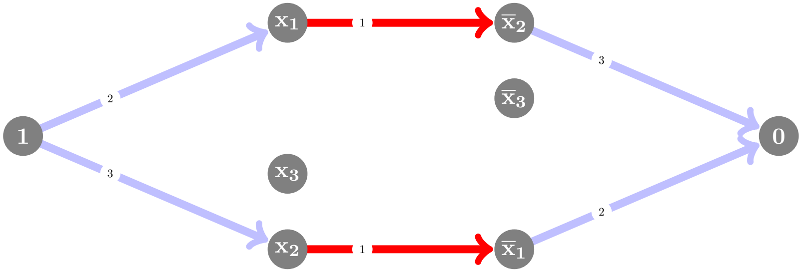
$$\mathbf{f} \quad = \color{red}{4\overline{\mathbf{x}}_1} + 6\overline{\mathbf{x}}_2 + 2\mathbf{x}_1\mathbf{x}_2 + 2\mathbf{x}_1\overline{\mathbf{x}}_3 + 4\mathbf{x}_2\mathbf{x}_3$$

Implication Networks



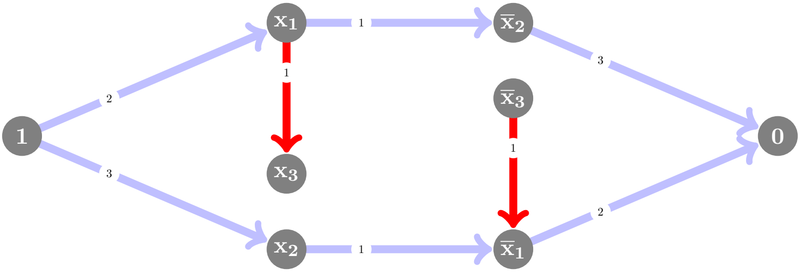
$$\mathbf{f} \quad = 4\overline{x}_1 + \color{red}{6\overline{x}_2} \color{black} + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3$$

Implication Networks



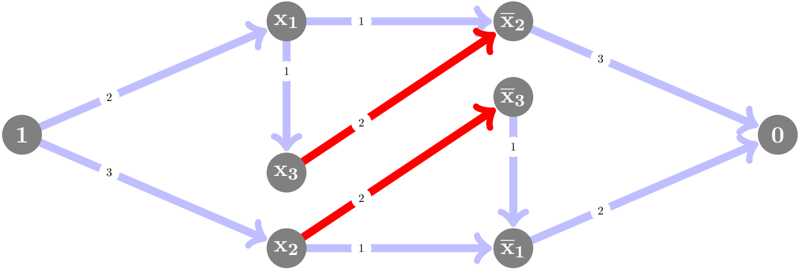
$$f = 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3$$

Implication Networks



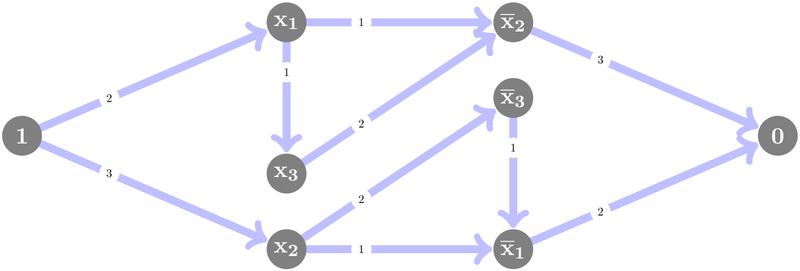
$$\mathbf{f} = 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3$$

Implication Networks



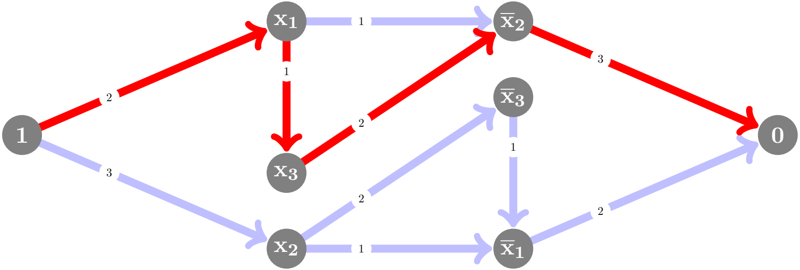
$$f \quad = 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + \color{red}{4x_2x_3}$$

Implication Networks



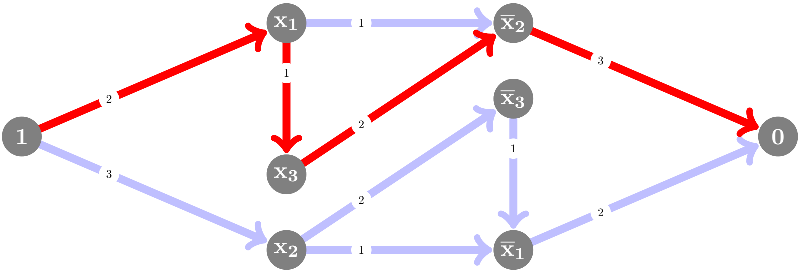
$$\mathbf{f} = 4\overline{\mathbf{x}}_1 + 6\overline{\mathbf{x}}_2 + 2\mathbf{x}_1\mathbf{x}_2 + 2\mathbf{x}_1\overline{\mathbf{x}}_3 + 4\mathbf{x}_2\mathbf{x}_3$$

Flows and Posiform Transformations



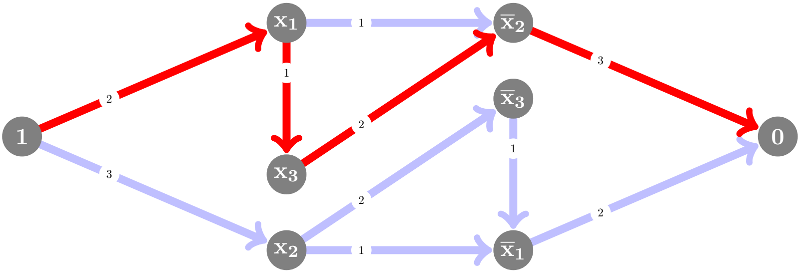
$$\mathbf{f} \quad = 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3$$

Flows and Posiform Transformations



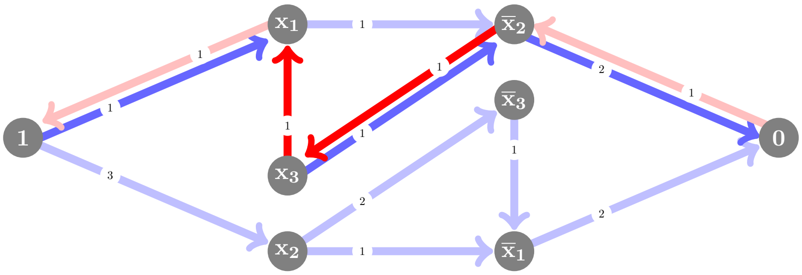
$$
\begin{aligned}
f &= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 \\
&\quad + \left(\overline{x}_1 + x_1\overline{x}_3 + x_3x_2 + \overline{x}_2\right)
\end{aligned}
$$

Flows and Posiform Transformations



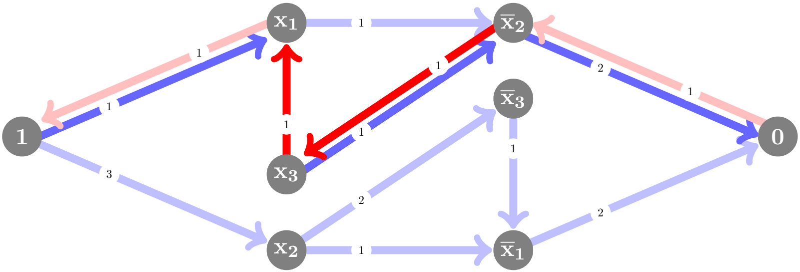
$$
\begin{aligned}
\mathbf{f} &= 4\bar{x}_1 + 6\bar{x}_2 + 2x_1x_2 + 2x_1\bar{x}_3 + 4x_2x_3 \\
&= 3\bar{x}_1 + 5\bar{x}_2 + 2x_1x_2 + x_1\bar{x}_3 + 3x_2x_3 \\
&\quad + (\bar{x}_1 + x_1\bar{x}_3 + x_3x_2 + \bar{x}_2) \\
&= 3\bar{x}_1 + 5\bar{x}_2 + 2x_1x_2 + x_1\bar{x}_3 + 3x_2x_3 \\
&\quad + (1 + \bar{x}_1x_3 + \bar{x}_3\bar{x}_2)
\end{aligned}
$$

Flows and Posiform Transformations



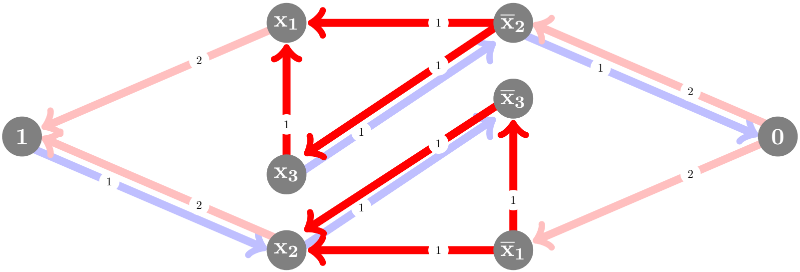
$$
\begin{aligned}
f &= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 \\
&\quad + (\overline{x}_1 + x_1\overline{x}_3 + x_3x_2 + \overline{x}_2) \\
&= 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 \\
&\quad + (1 + \overline{x}_1x_3 + \overline{x}_3\overline{x}_2) \\
&= 1 + 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 + \overline{x}_1x_3 + \overline{x}_3\overline{x}_2
\end{aligned}
$$

Flows and Posiform Transformations



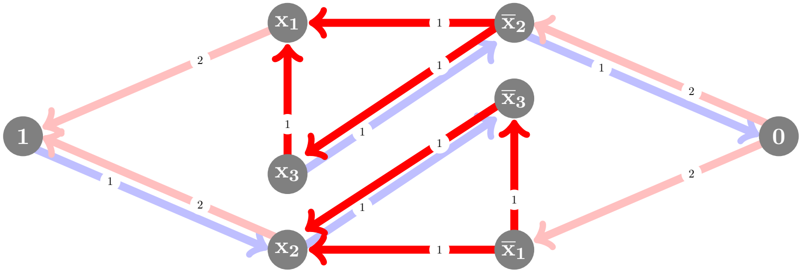
$$
\begin{aligned}
f &= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 1 + 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 + \overline{x}_1x_3 + \overline{x}_3\overline{x}_2
\end{aligned}
$$

Flows and Posiform Transformations



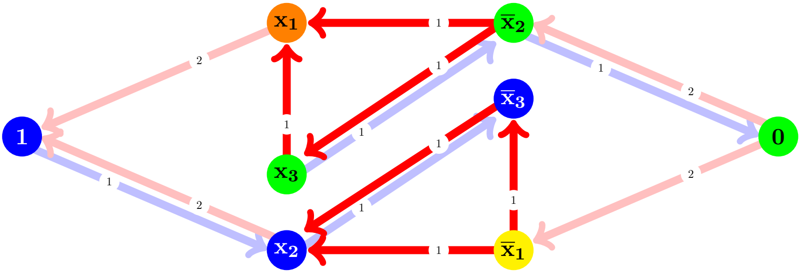
$$
\begin{aligned}
f &= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 1 + 3\overline{x}_1 + 5\overline{x}_2 + 2x_1x_2 + x_1\overline{x}_3 + 3x_2x_3 + \overline{x}_1x_3 + \overline{x}_3\overline{x}_2 \\
&= 4 + 2\overline{x}_2 + 2\overline{x}_1\overline{x}_2 + 2\overline{x}_1x_3 + 2x_2x_3 + 2\overline{x}_2\overline{x}_3
\end{aligned}
$$

Persistencies and Decompositions



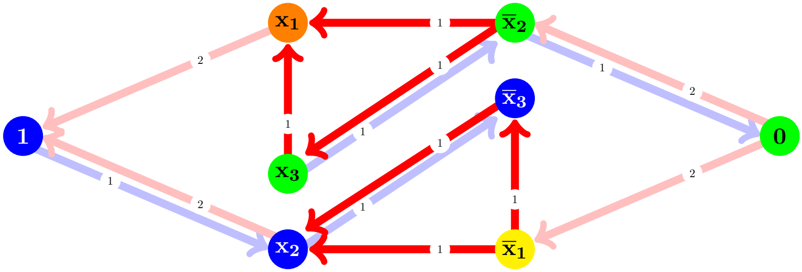
$$
\begin{aligned}
\mathbf{f} &= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3 \\
&= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 4 + 2\overline{x}_2 + 2\overline{x}_1\overline{x}_2 + 2\overline{x}_1x_3 + 2x_2x_3 + 2\overline{x}_2\overline{x}_3
\end{aligned}
$$

Persistencies and Decompositions



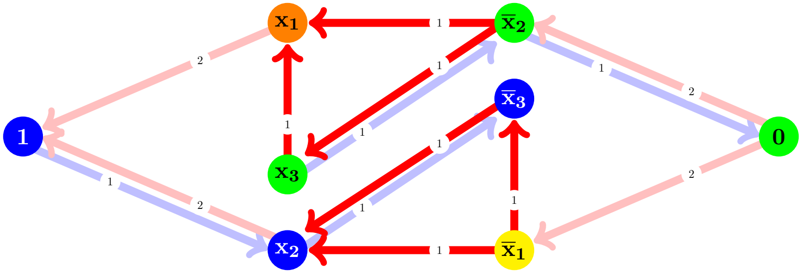
$$
\begin{aligned}
\mathbf{f} &= 10 - 2x_1 - 6x_2 + 2x_1x_2 - 2x_1x_3 + 4x_2x_3 \\
&= 4\overline{x}_1 + 6\overline{x}_2 + 2x_1x_2 + 2x_1\overline{x}_3 + 4x_2x_3 \\
&= 4 + 2\overline{x}_2 + 2\overline{x}_1\overline{x}_2 + 2\overline{x}_1x_3 + 2x_2x_3 + 2\overline{x}_2\overline{x}_3
\end{aligned}
$$

Persistencies and Decompositions



- Strong persistency: $x_2 = 1$, $x_3 = 0$
- Weak persistency: $x_1 = 1$ (or $x_1 = 0$)

Persistencies and Decompositions



- Strong persistency: $x_2 = 1$, $x_3 = 0$
- Weak persistency: $x_1 = 1$ (or $x_1 = 0$)

Persistencies and Decompositions

- Type I: $\mathbf{u} \in \mathbf{C} \implies \overline{\mathbf{u}} \notin \mathbf{C}$
- Type II: $\mathbf{u} \in \mathbf{C} \iff \overline{\mathbf{u}} \notin \mathbf{C}$
- Persistencies eliminate all type I strong components.
- Residual (type II) strong components decompose original problem into variable disjoint, independent subproblems.
- Submodular input (all quadratic coefficients are negative) results in an implication network consisting of two subnetworks on $\{x_1, ..., x_n\}$ and $\{\overline{x}_1, ...\overline{x}_n\}$, which are completely independent $\implies$ no type II strong component $\implies$ $\mathbf{C_2}(f) = \min f$!

Persistencies and Decompositions

- Type I: $\mathbf{u} \in \mathbf{C} \implies \overline{\mathbf{u}} \notin \mathbf{C}$
- Type II: $\mathbf{u} \in \mathbf{C} \iff \overline{\mathbf{u}} \notin \mathbf{C}$
- Persistencies eliminate all type I strong components.
- Residual (type II) strong components decompose original problem into variable disjoint, independent subproblems.
- Submodular input (all quadratic coefficients are negative) results in an implication network consisting of two subnetworks on $\{x_1, ..., x_n\}$ and $\{\overline{x}_1, ...\overline{x}_n\}$, which are completely independent $\implies$ no type II strong component $\implies$ $\mathbf{C_2}(f) = \min f$!

Persistencies and Decompositions

- Type I: $\mathbf{u} \in \mathbf{C} \implies \overline{\mathbf{u}} \notin \mathbf{C}$
- Type II: $\mathbf{u} \in \mathbf{C} \iff \overline{\mathbf{u}} \notin \mathbf{C}$
- Persistencies eliminate all type I strong components.
- Residual (type II) strong components decompose original problem into variable disjoint, independent subproblems.
- Submodular input (all quadratic coefficients are negative) results in an implication network consisting of two subnetworks on $\{x_1, ..., x_n\}$ and $\{\overline{x}_1, ...\overline{x}_n\}$, which are completely independent $\implies$ no type II strong component $\implies$ $\mathbf{C_2}(f) = \min f$!

Persistencies and Decompositions

- Type I: $\mathbf{u} \in \mathbf{C} \implies \overline{\mathbf{u}} \notin \mathbf{C}$
- Type II: $\mathbf{u} \in \mathbf{C} \iff \overline{\mathbf{u}} \notin \mathbf{C}$
- Persistencies eliminate all type I strong components.
- Residual (type II) strong components decompose original problem into variable disjoint, independent subproblems.
- Submodular input (all quadratic coefficients are negative) results in an implication network consisting of two subnetworks on $\{x_1, ..., x_n\}$ and $\{\overline{x}_1, ...\overline{x}_n\}$, which are completely independent $\implies$ no type II strong component $\implies$ $\mathbf{C_2}(f) = \min f$!

Persistencies and Decompositions

- Type I: $\mathbf{u} \in \mathbf{C} \implies \overline{\mathbf{u}} \notin \mathbf{C}$
- Type II: $\mathbf{u} \in \mathbf{C} \iff \overline{\mathbf{u}} \notin \mathbf{C}$
- Persistencies eliminate all type I strong components.
- Residual (type II) strong components decompose original problem into variable disjoint, independent subproblems.
- Submodular input (all quadratic coefficients are negative) results in an implication network consisting of two subnetworks on $\{x_1, ..., x_n\}$ and $\{\overline{x}_1, ...\overline{x}_n\}$, which are completely independent $\implies$ no type II strong component $\implies$ $\mathbf{C_2}(f) = \min f$!

Implication Networks and Persistencies

- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0,1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value $= \mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Implication Networks and Persistencies

- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0,1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value $= \mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Implication Networks and Persistencies

- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0,1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value $= \mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Implication Networks and Persistencies

- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0,1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value = $\mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Implication Networks and Persistencies

- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0, 1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value = $\mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Implication Networks and Persistencies

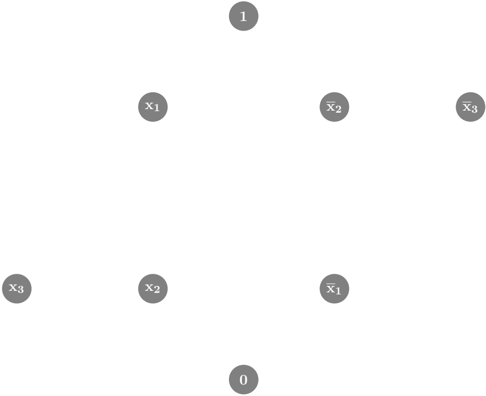
- There is a one-to-one correspondence between quadratic posiform transformations and flow-augmentations...
- For any binary assignment $x \in \{0,1\}^n$ there is a corresponding cut $(S, \overline{S})$ in the implication network of f such that $f(x) = cut(S, \overline{S})$.
- ... but, there are many other cuts in the implication networks!
- ..., therefore, the max-flow-min-cut value $= \mathbf{C_2}(f)$ is only a lower bound for f.
- There is a unique maximal subset of the variables which is fixed by persistencies (i.e., which participate in a type I strong component).
- There is a unique decomposition of the residual problem into variable disjoint smaller subproblems.

Outline

The Submodular Case

To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

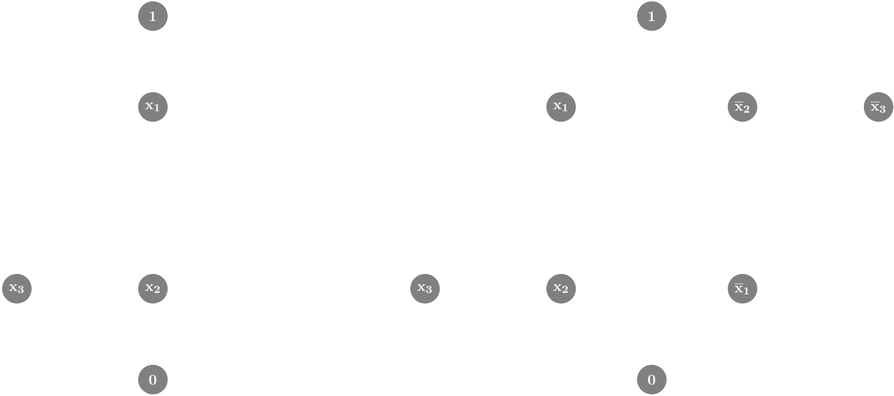
The Submodular Case



To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\mathbf{f} \quad = \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3}$$

The Submodular Case

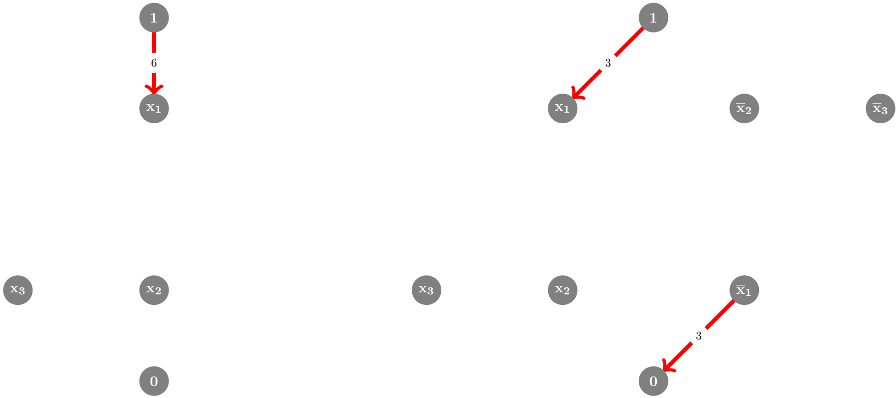


To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\mathbf{f} = 4 + 8\mathbf{x}_2 + 2\mathbf{x}_3 - 4\mathbf{x}_1\mathbf{x}_2 - 2\mathbf{x}_1\mathbf{x}_3 - 2\mathbf{x}_2\mathbf{x}_3$$

$$= -2 + 6\overline{\mathbf{x}}_1 + 6\mathbf{x}_2 + 2\mathbf{x}_3 + 4\mathbf{x}_1\overline{\mathbf{x}}_2 + 2\mathbf{x}_1\overline{\mathbf{x}}_3 + 2\mathbf{x}_2\overline{\mathbf{x}}_3$$

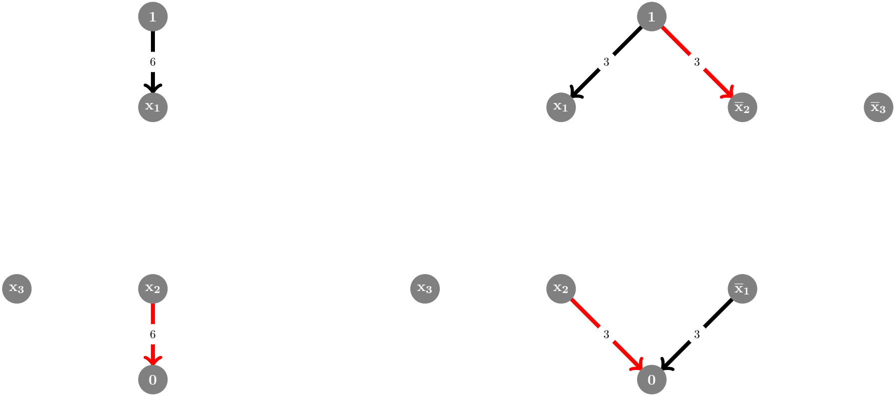
The Submodular Case



To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\begin{aligned}\mathbf{f} &= \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3} \\ &= \mathbf{-2 + {\color{red}6\overline{x}_1} + 6x_2 + 2x_3 + 4x_1\overline{x}_2 + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}\end{aligned}$$

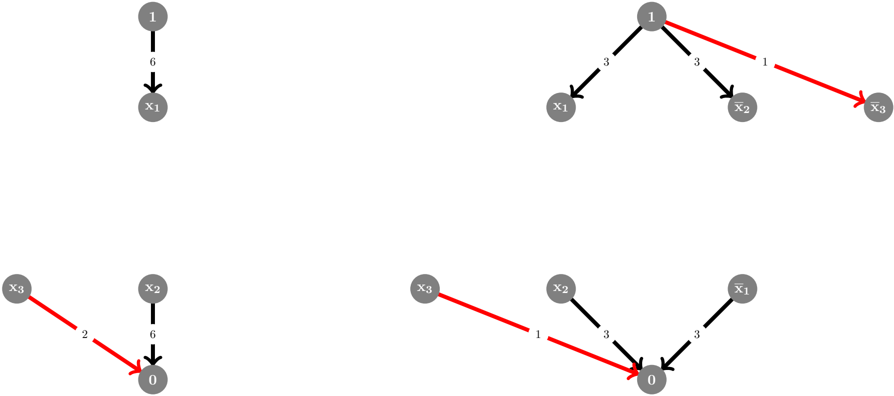
The Submodular Case



To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\begin{aligned}\mathbf{f} &= \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3} \\ &= \mathbf{-2 + 6\overline{x}_1 + {\color{red}6x_2} + 2x_3 + 4x_1\overline{x}_2 + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}\end{aligned}$$

The Submodular Case

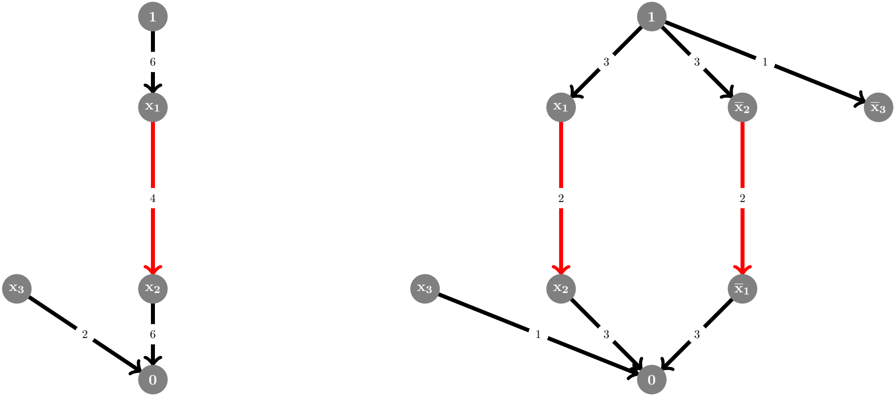


To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\mathbf{f} = \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3}$$

$$= \mathbf{-2 + 6\overline{x}_1 + 6x_2 + {\color{red}2x_3} + 4x_1\overline{x}_2 + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}$$

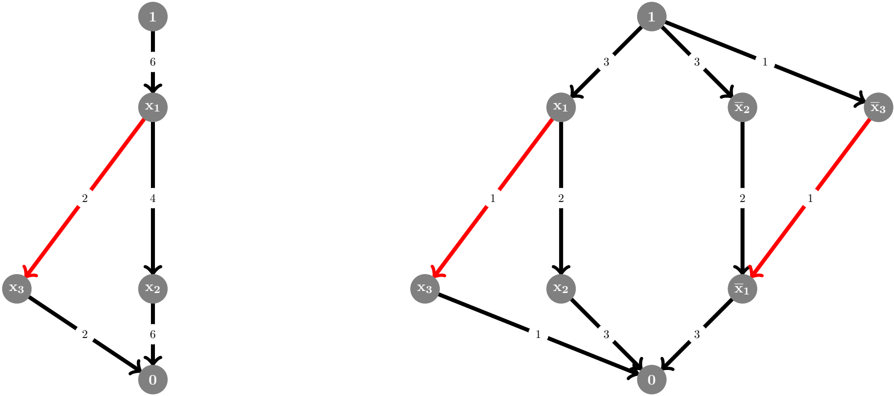
The Submodular Case



To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\begin{aligned}\mathbf{f} &= \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3} \\ &= \mathbf{-2 + 6\overline{x}_1 + 6x_2 + 2x_3 + {\color{red}4x_1\overline{x}_2} + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}\end{aligned}$$

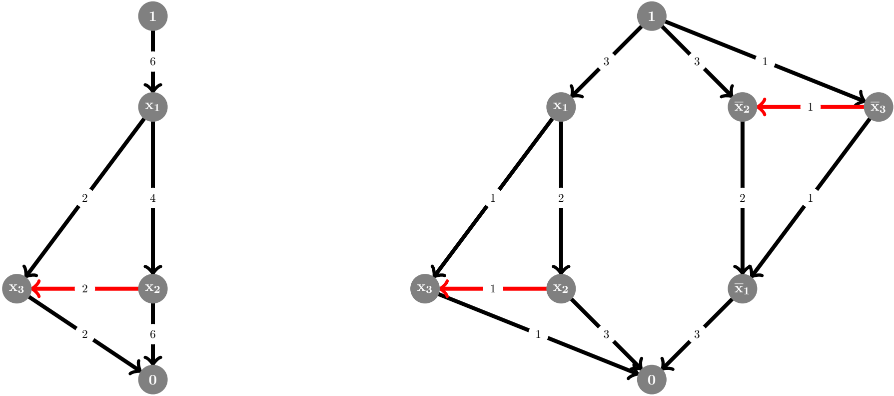
The Submodular Case



To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\begin{aligned} \mathbf{f} &= \mathbf{4} + \mathbf{8x_2} + \mathbf{2x_3} - \mathbf{4x_1x_2} - \mathbf{2x_1x_3} - \mathbf{2x_2x_3} \\ &= -\mathbf{2} + \mathbf{6\overline{x}_1} + \mathbf{6x_2} + \mathbf{2x_3} + \mathbf{4x_1\overline{x}_2} + {\color{red}\mathbf{2x_1\overline{x}_3}} + \mathbf{2x_2\overline{x}_3} \end{aligned}$$

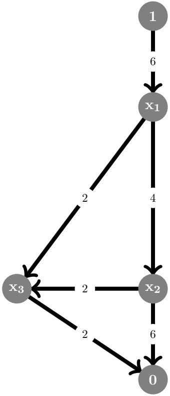
The Submodular Case

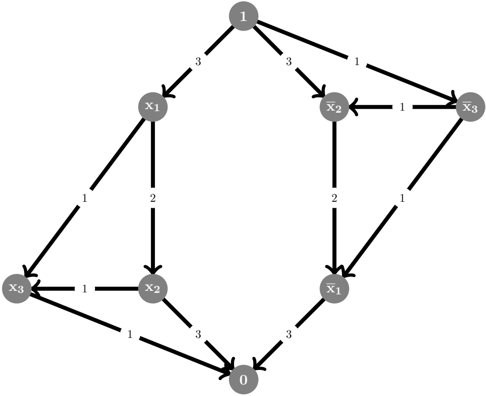


To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$\begin{aligned}\mathbf{f} &= \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3} \\ &= \mathbf{-2 + 6\overline{x}_1 + 6x_2 + 2x_3 + 4x_1\overline{x}_2 + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}\end{aligned}$$

The Submodular Case





To a submodular QPBF f we can associate both a graph-cut network G_f and an implication network N_f:

$$
\begin{aligned}
\mathbf{f} &= \mathbf{4 + 8x_2 + 2x_3 - 4x_1x_2 - 2x_1x_3 - 2x_2x_3} \\
&= \mathbf{-2 + 6\overline{x}_1 + 6x_2 + 2x_3 + 4x_1\overline{x}_2 + 2x_1\overline{x}_3 + 2x_2\overline{x}_3}
\end{aligned}
$$

Outline

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

Components of the Algorithm

The **purpose** of the preprocessing algorithm is to **fix** some of the variables at their optimum values and **decompose** the remaining problem into several smaller problems which do not share variables, **in strongly polynomial time**.

- Build implication network
- Compute maximum flow; fix variables by persistency (increase capacities of some arcs)
- Probe remaining variables and repeat all of the above as long as there is some change.
- Output remaining strong components, if any.

If the input QPBF is submodular, then the above procedure will fix all the variables at their optimal values in the first round, without any probing.

Outline

Via Minimization in VLSI Design

		Percentage of Variables Fixed by					
Problem[2]	n	Persistency		Probing		**ALL TOOLS**	**Time (sec)**
		(strong)	(weak)	(forc)	(equal)		
via.c1y	829	93.6%	6.4%	0%	0%	**100%**	**0.03**
via.c2y	981	94.7%	5.3%	0%	0%	**100%**	**0.06**
via.c3y	1328	94.6%	5.4%	0%	0%	**100%**	**0.09**
via.c4y	1367	96.4%	3.6%	0%	0%	**100%**	**0.09**
via.c5y	1203	93.1%	6.9%	0%	0%	**100%**	**0.08**
via.c1n	828	57.4%	9.6%	32.4%	0.6%	**100%**	**0.49**
via.c2n	980	12.4%	4.4%	83.1%	0.1%	**100%**	**7.14**
via.c3n	1327	6.8%	5.7%	87.3%	0.2%	**100%**	**18.17**
via.c4n	1366	11.1%	1.3%	87.6%	0%	**100%**	**23.08**
via.c5n	1202	3.4%	1.4%	95.0%	0.2%	**100%**	**17.13**

[2] S. Homer and M. Peinado. Design and performance of parallel and distributed approximation algorithms for maxcut. Journal of Parallel and Distributed Computing 46 (1997) 48-61.

Vertex Cover in Planar Graphs

	Averages for 100 graphs in each of the 4 groups			
	Variables Fixed (%)		Time (sec)	
n	A. D. N.[3]	**QUBO**[4]	A. D. N.[2]	**QUBO**[3]
1000	68.4	**100**	4.06	**0.05**
2000	67.4	**100**	12.24	**0.16**
3000	65.5	**100**	30.90	**0.27**
4000	62.7	**100**	60.45	**0.53**

[3] Alber, Dorn, Niedermeier. Experimental evaluation of a tree decomposition based algorithm for vertex cover on planar graphs. Discrete Applied Mathematics 145 (2005) 219-231; 750 GHz, Linux PC, 720 MB

[4] Pentium 4, 2.8 GHz, Windows XP, 512 MB

Jumbo Vertex Cover in Planar Graphs

Vertices	Computing Times (min)[5]		
	Planar Density		
	10%	50%	90%
50,000	0.7	2.3	0.9
100,000	2.9	10.2	3.9
250,000	19.5	69.8	26.3
500,000	79.3	277.3	106.9

QUBO fixed all variables for all problems!

[5] Averages over 3 experiments on a Xeon 3.06 GHz, XP, 3.5 GB RAM.

One Dimensional Ising Models

σ	Number of Spins	Average Computing Time (s)		
		Branch, Cut & Price[6]	Biq Maq[5]	**QUBO**[7]
2.5	100	699	68	**1**
	150	92 079	388	**3**
	200	N/A	993	**9**
	250	N/A	6 567	**14**
	300	N/A	34 572	**21**
3.0	100	256	59	**1**
	150	13 491	293	**2**
	200	61 271	1 034	**3**
	250	55 795	3 594	**4**
	300	55 528	8 496	**5**

[6] F. Rendl, G. Rinaldi, A. Wiegele. (2007). Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations.

[7] ALL problems were solved by QUBO.

Larger One Dimensional Ising Models

σ	n	Average of 3 Problems: Variables not fixed	Average of 3 Problems: QUBO Time (s)[8]
2.5	500	**5**	**13**
	750	**22**	**30**
	1000	**24**	**53**
	1250	**20**	**81**
	1500	**32**	**124**
3.0	500	**0**	**4**
	750	**0**	**12**
	1000	**0**	**23**
	1250	**0**	**37**
	1500	**0**	**59**

[8] Pentium M, 1.6 GHz 760 MB RAM

Outline

References

- Hammer, P.L. Some network flow problems solved with pseudo-Boolean programming. *Operations Research* **13** (1965) 388-399.
- Hammer, P.L. and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas.* (Springer-Verlag, Berlin, Heidelberg, New York, 1968.)
- Hammer, P.L., P. Hansen and B. Simeone. Roof duality, complementation and persistency in quadratic $0-1$ optimization. *Mathematical Programming* **28** (1984), pp. 121-155.
- Boros E. and P.L. Hammer. A max-flow approach to improved roof-duality in quadratic $0-1$ minimization. RUTCOR Research Report RRR 15-1989, RUTCOR, March 1989.
- Boros, E., Y. Crama, and P.L. Hammer. Upper bounds for quadratic $0-1$ maximization. *Operations Research Letters*, **9** (1990), 73-79,

References Cont'd

- Billionnet, A. and A. Sutter. Persistency in quadratic $0-1$ optimization. *Math. Programming* **54** (1992), no. 1, Ser. A, pp. 115–119.
- Boros, E., Y. Crama and P.L. Hammer. Chvátal cuts and odd cycle inequalities in quadratic $0-1$ optimization. *SIAM Journal on Discrete Mathematics*, **5** (1992), 163-177.
- Boros, E. and P.L. Hammer, Pseudo-Boolean Optimization, *Discrete Applied Mathematics*, **123** (2002) 155–225.
- Boros, E., P.L. Hammer and G. Tavares. Preprocessing for unconstrained quadratic binary programming. RUTCOR Research Report, 10-2006.
- Boros, E., P.L. Hammer, G. Tavares, and R. Sun. A max-flow approach to improved lower bounds for quadratic $0-1$ minimization. *Discrete Optimization*, **5/2** (2007) 501-529.

THANK YOU!