Solving SAT with a Quantum Annealer: Foundations, Encodings and a Preliminary Report

Roberto Sebastiani
DISI, University of Trento, Italy

joint work with:

William Macready, Zhengbing Bian, Fabian Chudak, Aidan Roy
formerly at D-Wave Systems Inc., Burnaby, Canada

Stefano Varotti
formerly at DISI, University of Trento, Italy

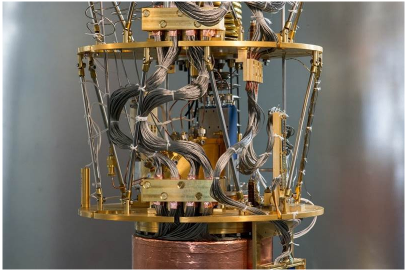
Based on the paper:
Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, Stefano Varotti
"Solving SAT (and MaxSAT) with a Quantum Annealer: Foundations, Encodings and a Preliminary Report".
Information and Computation, vol 275, December 2020.

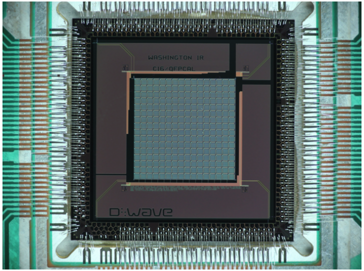
https://doi.org/10.1016/j.ic.2020.104609

Outline

Outline

D-Wave's Quantum Annealers





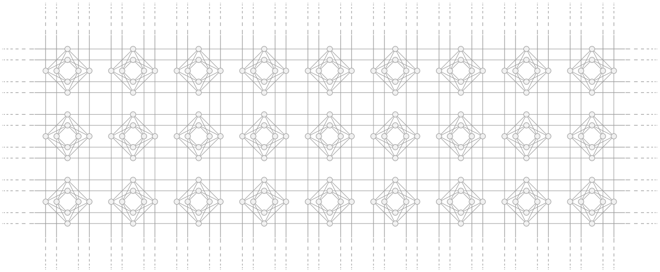
© D-Wave Systems Inc.

- specialized chips using quantum effects (superposition, tunneling) to find minimum energy configurations over binary variables (qubits) $z_i \in \{-1, 1\}$ (aka Ising Hamiltonian):

$$H(\underline{\mathbf{z}}) \stackrel{\text{def}}{=} \sum_{i \in V} h_i z_i + \sum_{\langle i,j \rangle \in E} J_{ij} z_i z_j$$

- $\langle V, E \rangle$ connection graph of the qubits
(e.g. 16×16 grid of 8-qubit tiles in Chimera architecture)

D-Wave's Quantum Annealers

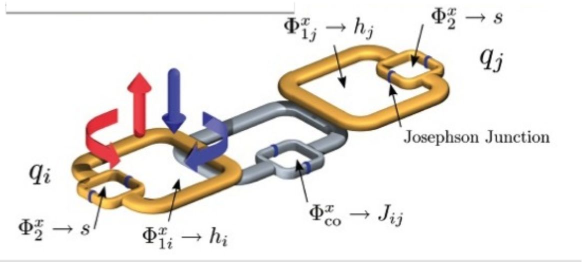


- specialized chips using quantum effects (superposition, tunneling) to find minimum energy configurations over binary variables (qubits) $z_i \in \{-1, 1\}$ (aka Ising Hamiltonian):

$$H(\underline{\mathbf{z}}) \stackrel{\text{def}}{=} \sum_{i \in V} h_i z_i + \sum_{\langle i,j \rangle \in E} J_{ij} z_i z_j$$

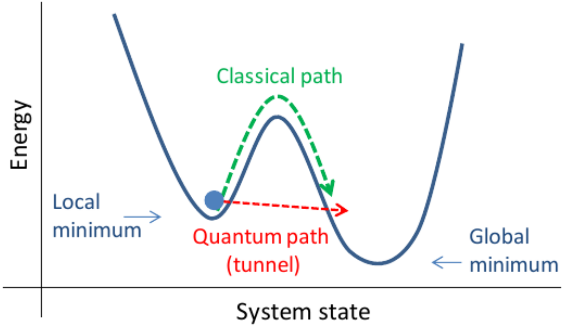
- $\langle V, E \rangle$ connection graph of the qubits
  (e.g. 16×16 grid of 8-qubit tiles in Chimera architecture)

D-Wave's Quantum Annealers



- qubits are super-conducting rings
  - qubit value $\{-1, 1\}$ is the direction of circulating current
  - superposition of clockwise and counterclockwise (!)
  - we can apply bias h_i to each qubit z_i, and a coupling J_{ij} to connected pairs of qubits $\langle z_i, z_j \rangle$
- Quantum system transition: when measuring (sampling), system ends in low energy state
  $H_{final}(\mathbf{z}) \stackrel{\text{def}}{=} \sum_{i \in V} h_i z_i + \sum_{\langle i,j \rangle \in E} J_{ij} z_i z_j$
  (ideally, a minimum-energy state)

D-Wave's Quantum Annealers

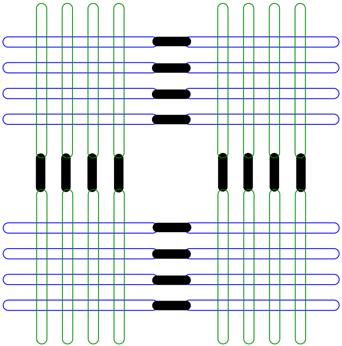


- $H(\underline{\mathbf{z}})$ defines a problem by inducing an energy landscape s.t. the lowest point is the best solution
  - "classical" annealing can only walk over this landscape
  - Q.A. uses quantum effects (tunneling) to go through the hills

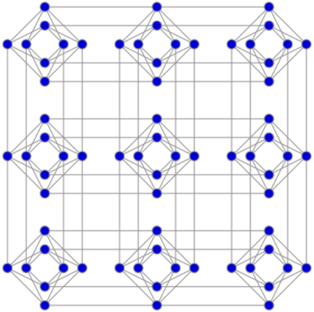
$\Longrightarrow$ can reach global minima

- works well with tall & narrow energy barriers separating minima

Chimera Tale-based architecture



Coupling Schema



Connection Graph

- 8-qubit “tiles”
- low inter-tile connections
- 6 couplings/qubit overall
- no cliques

Outline

1. Background: D-Wave's Quantum Annealers
2. Motivations & Goals
3. Solving SAT with Quantum Annealers: Foundations
   - Penalty Functions
   - Encoding SAT into Penalty Functions
   - Some Issues
4. Preliminary Empirical Results

General Motivations & Goals

NP-complete problems

- Problems intrinsically computationally hard for classic computers
  - No polynomial-time algorithm believed to exist
    $\Longrightarrow$ CPU time blows up wrt. problem size in worst case
  - NP-complete problems can be polynomially reduced to each other

  $\Longrightarrow$ [s]he who polynomially solved one, would polynomially solve all
  - Huge amount of industrial applications (HW/SW synthesis & verification, cryptography & security, scheduling, planning, ...)

$\Longrightarrow$ Any major advance in efficiently solving NP-complete problems will have a huge application impact.

Big research question

Can quantum technologies bring to one such major advance?

Holy Grail: Quantum Supremacy on NP-complete problems

Solve via quantum devices (useful) NP-complete problems that no classical computer can solve in any feasible amount of time

General Motivations & Goals

NP-complete problems

- Problems intrinsically computationally hard for classic computers
  - No polynomial-time algorithm believed to exist
    $\Longrightarrow$ CPU time blows up wrt. problem size in worst case
  - NP-complete problems can be polynomially reduced to each other

  $\Longrightarrow$ [s]he who polynomially solved one, would polynomially solve all
  - Huge amount of industrial applications (HW/SW synthesis & verification, cryptography & security, scheduling, planning, ...)

$\Longrightarrow$ Any major advance in efficiently solving NP-complete problems will have a huge application impact.

Big research question

Can quantum technologies bring to one such major advance?

Holy Grail: Quantum Supremacy on NP-complete problems

Solve via quantum devices (useful) NP-complete problems that no classical computer can solve in any feasible amount of time

General Motivations & Goals

NP-complete problems

- Problems intrinsically computationally hard for classic computers
  - No polynomial-time algorithm believed to exist
    $\Longrightarrow$ CPU time blows up wrt. problem size in worst case
  - NP-complete problems can be polynomially reduced to each other

  $\Longrightarrow$ [s]he who polynomially solved one, would polynomially solve all
  - Huge amount of industrial applications (HW/SW synthesis & verification, cryptography & security, scheduling, planning, ...)

$\Longrightarrow$ Any major advance in efficiently solving NP-complete problems will have a huge application impact.

Big research question

Can quantum technologies bring to one such major advance?

Holy Grail: Quantum Supremacy on NP-complete problems

Solve via quantum devices (useful) NP-complete problems that no classical computer can solve in any feasible amount of time

Our Goals

Our Goal: SAT-to-QUBO

SAT solving via Quantum Annealing (Q.A.):
find a way to encode Boolean formulas into an Ising Hamiltonian (aka QUBO) to be fed to and minimized by the Q.A.

- effectively: the QUBO fits into and can be solved by the Q.A.
- efficiently: the encoding process should be resource efficient

Focus: small-but-hard problems

Problems which are small enough to fit into the Q.A. architecture,
but hard enough to be challenging for standard SAT solvers

- Ex: problems from cryptanalysis, HW arithmetical circuits, parity/cardinality constraints ...

Our Goals

Our Goal: SAT-to-QUBO

SAT solving via Quantum Annealing (Q.A.):
find a way to encode Boolean formulas into an Ising Hamiltonian (aka QUBO) to be fed to and minimized by the Q.A.

- effectively: the QUBO fits into and can be solved by the Q.A.
- efficiently: the encoding process should be resource efficient

Focus: small-but-hard problems

Problems which are small enough to fit into the Q.A. architecture,
but hard enough to be challenging for standard SAT solvers

- Ex: problems from cryptanalysis, HW arithmetical circuits, parity/cardinality constraints ...

Outline

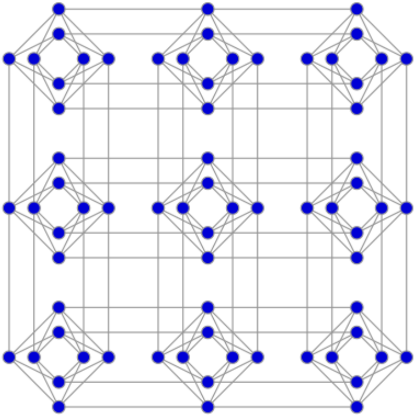
Outline

Quantum Annealers from a CS Perspective

- A "magic" blackbox device solving a (very-restricted form of) Quadratic Unconstrained Binary Optimization (QUBO) problem:

$$argmin_{[\underline{\mathbf{z}}\in\{-1,1\}^{|V|}]} \left(\theta_0 + \sum_{i\in V} \theta_i z_i \;+\; \sum_{\langle i,j\rangle\in E} \theta_{ij} z_i z_j\right)$$

  - Each $z_i \in \{-1, 1\}$ is the value of the i-th qubit
  - programmable numerical values:
    $\theta_0 \in (-\infty, +\infty)$ (offset),
    $\theta_i \in [-2, 2]$ (biases),
    $\theta_{ij} \in [-1, 1]$ (couplings)
    ($\theta_{ij} = 0$ if $\langle i, j\rangle \notin E$)
  - $\langle V, E\rangle$ is the connectivity graph:
    - e.g. $N \times N$ grid of 8-qubit clusters (tiles) in Chimera architecture
    - e.g.(D-Wave 2000Q): $16 \times 16 \times 8 = 2048$ qubits



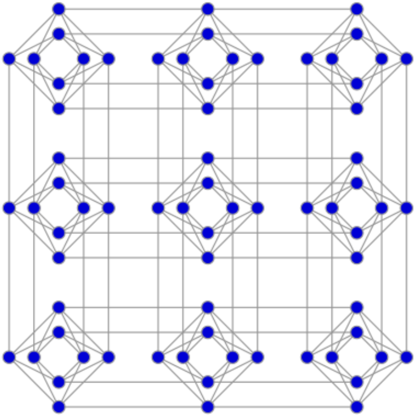
- NP-complete
- $\langle V, E\rangle$ undirected and sparse: each z_i has ≤ 6 connections

Quantum Annealers from a CS Perspective

- A "magic" blackbox device solving a (very-restricted form of) Quadratic Unconstrained Binary Optimization (QUBO) problem:

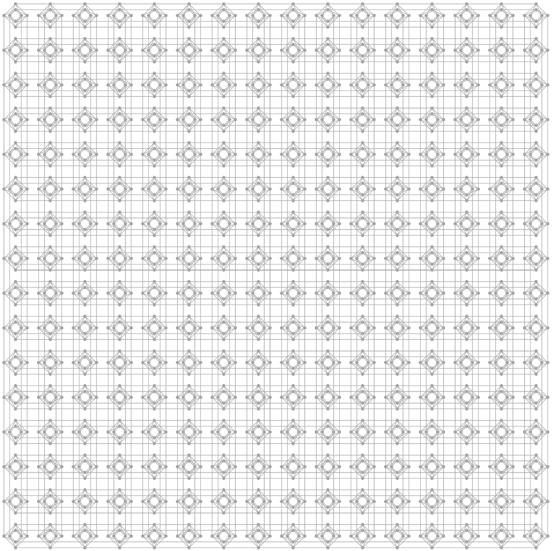
$$argmin_{[\underline{\mathbf{z}} \in \{-1,1\}^{|V|}]} P(\underline{\mathbf{z}}|\underline{\boldsymbol{\theta}}) \text{ ``penalty function''}$$

  - Each $z_i \in \{-1, 1\}$ is the value of the i-th qubit
  - programmable numerical values:
    $\theta_0 \in (-\infty, +\infty)$ (offset),
    $\theta_i \in [-2, 2]$ (biases),
    $\theta_{ij} \in [-1, 1]$ (couplings)
    ($\theta_{ij} = 0$ if $\langle i, j \rangle \notin E$)
  - $\langle V, E \rangle$ is the connectivity graph:
    - e.g. $N \times N$ grid of 8-qubit clusters (tiles) in Chimera architecture
    - e.g.(D-Wave 2000Q): $16 \times 16 \times 8 = 2048$ qubits



- NP-complete
- $\langle V, E \rangle$ undirected and sparse: each z_i has ≤ 6 connections

The D-Wave 2000Q 2046-qubit topology



Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^{k}]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \begin{cases} = 0 & \text{if } \underline{\mathbf{x}} \models F \\ > 0 & \text{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} \; P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\boldsymbol{\theta}}) \begin{cases} = 0 & \text{if } \underline{\mathbf{x}} \models F \\ > 0 & \text{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\boldsymbol{\theta}}) \begin{cases} = 0 & \text{if } \underline{\mathbf{x}} \models F \\ > 0 & \text{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \begin{cases} = 0 & \textit{if } \underline{\mathbf{x}} \models F \\ > 0 & \textit{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\boldsymbol{\theta}) \begin{cases} = 0 & \text{if } \underline{\mathbf{x}} \models F \\ \geq g_{min} & \text{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} \; P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \begin{cases} = 0 & \textit{if } \underline{\mathbf{x}} \models F \\ \geq g_{min} & \textit{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\iff$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \begin{cases} = 0 & \text{if } \underline{\mathbf{x}} \models F \\ \geq g_{min} & \text{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

$\implies$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\implies$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

Problems

- Goal (SAT-to-QUBO): find a general procedure to encode a Boolean formula $F(\underline{\mathbf{x}})$ into a such model

$\Longleftrightarrow$ find a variable placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \mapsto \underline{\mathbf{z}}$ and values for the θ's s.t.

$$min_{[\underline{\mathbf{a}} \in \{-1,1\}^k]} P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) \begin{cases} = 0 & \textit{if } \underline{\mathbf{x}} \models F \\ = g_{min} & \textit{otherwise} \end{cases} \quad \{\top, \bot\} \stackrel{\text{def}}{=} \{1, -1\}$$

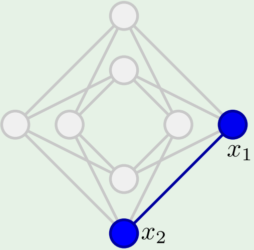
$\Longrightarrow$ if the Q.A. returns 0, then $F(x)$ is satisfiable

- P_F called Exact Penalty Function
- Critical issues:
  - limited number of qubits (currently 2048)
  - low number of connections between qubits (currently 6), no cliques
  - sensitivity: a big-enough gap $g_{min} > 0$ necessary
  - problem is over-constrained:
    - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels
    - $O(|\underline{\mathbf{z}}|)$ degrees of freedom θ_i, θ_{ij}

$\Longrightarrow$ need ancillary Boolean variables $\underline{\mathbf{a}}$ s.t. $\underline{\mathbf{z}} = \underline{\mathbf{x}} \cup \underline{\mathbf{a}}$

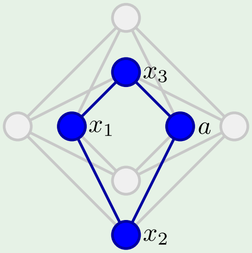
Simple examples of penalty functions

1. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} (x_1 \leftrightarrow x_2)$, then $P_F(\underline{\mathbf{x}}|\underline{\theta}) \stackrel{\text{def}}{=} 1 - x_1 x_2$
   $\Longrightarrow$ $\theta_0 = 1$, $\theta_1 = \theta_2 = 0$, $\theta_{12} = -1$, $g_{min} = 2$
   - exact
   - no ancilla needed
2. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \wedge x_2)$, then
   $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$
   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - one ancilla a needed
3. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \oplus x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) =$
   $5+x_3+a_2-a_3+x_1a_1-x_1a_2-x_1a_3-x_2a_1-x_2a_2-x_2a_3+x_3a_2-x_3a_3$
   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - 3 ancillas a_1, a_2, a_3 needed



Simple examples of penalty functions

1. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} (x_1 \leftrightarrow x_2)$, then $P_F(\underline{\mathbf{x}}|\underline{\theta}) \stackrel{\text{def}}{=} 1 - x_1 x_2$
   $\Longrightarrow$ $\theta_0 = 1$, $\theta_1 = \theta_2 = 0$, $\theta_{12} = -1$, $g_{min} = 2$
   - exact
   - no ancilla needed
2. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \wedge x_2)$, then
   $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1 x_2 - x_1 x_3 - x_2 a - x_3 a$
   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - one ancilla a needed
3. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \oplus x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) =$
   $5+x_3+a_2-a_3+x_1 a_1-x_1 a_2-x_1 a_3-x_2 a_1-x_2 a_2-x_2 a_3+x_3 a_2-x_3 a_3$
   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - 3 ancillas a_1, a_2, a_3 needed



Simple examples of penalty functions

1. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} (x_1 \leftrightarrow x_2)$, then $P_F(\underline{\mathbf{x}}|\underline{\theta}) \stackrel{\text{def}}{=} 1 - x_1 x_2$

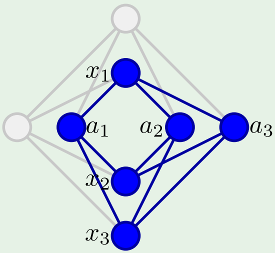
   $\Longrightarrow \theta_0 = 1, \theta_1 = \theta_2 = 0, \theta_{12} = -1, g_{min} = 2$
   - exact
   - no ancilla needed
2. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \wedge x_2)$, then

   $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$

   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - one ancilla a needed
3. if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (x_1 \oplus x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) =$

   $5 + x_3 + a_2 - a_3 + x_1a_1 - x_1a_2 - x_1a_3 - x_2a_1 - x_2a_2 - x_2a_3 + x_3a_2 - x_3a_3$

   $\Longrightarrow$ ..., $g_{min} = 2$
   - non-exact
   - 3 ancillas a_1, a_2, a_3 needed



Normal Penalty Functions & Normalization

Definition

A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}})$ is normal if $|\theta_i| = 2$ for at least one θ_i or $|\theta_{ij}| = 1$ for at least one θ_{ij}.

Examples

1. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$ is normal (e.g. $\theta_{x_1x_3} = -1$)
2. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{4} - \frac{1}{4}x_1 - \frac{1}{4}x_2 + \frac{1}{2}x_3 + \frac{1}{4}x_1x_2 - \frac{1}{2}x_1x_3 - \frac{1}{2}x_2a - \frac{1}{2}x_3a$ is not normal
3. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 5 - x_1 - x_2 + 2x_3 + x_1x_2 - 2x_1x_3 - 2x_2a - 2x_3a$ is not correct (out of ranges)

- To maximize g_{min}, use normal penalty functions
  $\Longrightarrow$ exploit the full range of the $\underline{\theta}$ parameters.
- A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ can be normalized by multiplying all its coefficients by a normalization factor:

$$c \stackrel{\text{def}}{=} \min\left\{\min_i\left(\frac{2}{|\theta_i|}\right), \min_{\langle ij\rangle}\left(\frac{1}{|\theta_{ij}|}\right)\right\}.$$

- E.g., normalize (2) by multiplying all coefficients by 2.

Normal Penalty Functions & Normalization

Definition

A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ is normal if $|\theta_i| = 2$ for at least one θ_i or $|\theta_{ij}| = 1$ for at least one θ_{ij}.

Examples

1. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$ is normal (e.g. $\theta_{x_1x_3} = -1$)
2. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{4} - \frac{1}{4}x_1 - \frac{1}{4}x_2 + \frac{1}{2}x_3 + \frac{1}{4}x_1x_2 - \frac{1}{2}x_1x_3 - \frac{1}{2}x_2a - \frac{1}{2}x_3a$ is not normal
3. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 5 - x_1 - x_2 + 2x_3 + x_1x_2 - 2x_1x_3 - 2x_2a - 2x_3a$ is not correct (out of ranges)

- To maximize g_{min}, use normal penalty functions $\Longrightarrow$ exploit the full range of the $\underline{\theta}$ parameters.
- A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ can be normalized by multiplying all its coefficients by a normalization factor:

$$c \stackrel{\text{def}}{=} \min\left\{\min_i\left(\frac{2}{|\theta_i|}\right), \min_{\langle ij\rangle}\left(\frac{1}{|\theta_{ij}|}\right)\right\}.$$

- E.g., normalize (2) by multiplying all coefficients by 2.

Normal Penalty Functions & Normalization

Definition

A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta})$ is normal if $|\theta_i| = 2$ for at least one θ_i or $|\theta_{ij}| = 1$ for at least one θ_{ij}.

Examples

1. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$ is normal (e.g. $\theta_{x_1x_3} = -1$)
2. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \frac{5}{4} - \frac{1}{4}x_1 - \frac{1}{4}x_2 + \frac{1}{2}x_3 + \frac{1}{4}x_1x_2 - \frac{1}{2}x_1x_3 - \frac{1}{2}x_2a - \frac{1}{2}x_3a$ is not normal
3. $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = 5 - x_1 - x_2 + 2x_3 + x_1x_2 - 2x_1x_3 - 2x_2a - 2x_3a$ is not correct (out of ranges)

- To maximize g_{min}, use normal penalty functions $\Longrightarrow$ exploit the full range of the $\underline{\theta}$ parameters.
- A penalty function $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta})$ can be normalized by multiplying all its coefficients by a normalization factor:

$$c \stackrel{\text{def}}{=} \min\left\{ \min_i \left(\frac{2}{|\theta_i|} \right), \min_{\langle ij \rangle} \left(\frac{1}{|\theta_{ij}|} \right) \right\}.$$

- E.g., normalize (2) by multiplying all coefficients by 2.

Properties of penalty functions: NPN-Equivalence

Let $F(\underline{\mathbf{x}})$ be a Boolean function, and let $F^*(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} F(x_1, ..., x_{i-1}, \neg x_i, x_{i+1}, ..., x_n)$ for some index i.
Suppose also that $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ is a penalty function for $F(\underline{\mathbf{x}})$ under some placement.
Then $P_{F^*}(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}^*)$, where $\underline{\theta}^*$ is s.t., for every $z, z' \in \underline{\mathbf{x}}, \underline{\mathbf{a}}$:

$$\theta^*_z = \begin{cases} -\theta_z & \textit{if } z = x_i \\ \theta_z & \textit{otherwise} \end{cases} \qquad \theta^*_{zz'} = \begin{cases} -\theta_{zz'} & \textit{if } z = x_i \textit{ or } z' = x_i \\ \theta_{zz'} & \textit{otherwise} \end{cases}$$

Example

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\ \ x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$
if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\neg x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} + \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 - \frac{1}{2}x_1x_2 + x_1x_3 - x_2a - x_3a$

Two Boolean functions $F(\underline{\mathbf{x}}), F^*(\underline{\mathbf{x}})$ that become equivalent by permuting or negating some of their input variables $\underline{\mathbf{x}}$ are called NPN-equivalent.
$\Longrightarrow$ given a penalty function for F, computing one for F^* is trivial.

$x_3 \leftrightarrow (x_1 \wedge x_2)$, $\neg x_3 \leftrightarrow (\neg x_1 \wedge \neg x_2)$, $x_3 \leftrightarrow (x_1 \vee x_2)$, $x_3 \leftrightarrow (x_1 \rightarrow x_2)$, ... are NPN-equivalent

Properties of penalty functions: NPN-Equivalence

Let $F(\underline{\mathbf{x}})$ be a Boolean function, and let $F^*(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} F(x_1, ..., x_{i-1}, \neg x_i, x_{i+1}, ..., x_n)$ for some index i.
Suppose also that $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ is a penalty function for $F(\underline{\mathbf{x}})$ under some placement.
Then $P_{F^*}(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}^*)$, where $\underline{\theta}^*$ is s.t., for every $z, z' \in \underline{\mathbf{x}}, \underline{\mathbf{a}}$:

$$\theta^*_z = \begin{cases} -\theta_z & \text{if } z = x_i \\ \theta_z & \text{otherwise} \end{cases} \qquad \theta^*_{zz'} = \begin{cases} -\theta_{zz'} & \text{if } z = x_i \text{ or } z' = x_i \\ \theta_{zz'} & \text{otherwise} \end{cases}$$

Example

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\ \ x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\neg x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} + \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 - \frac{1}{2}x_1x_2 + x_1x_3 - x_2a - x_3a$

Two Boolean functions $F(\underline{\mathbf{x}}), F^*(\underline{\mathbf{x}})$ that become equivalent by permuting or negating some of their input variables $\underline{\mathbf{x}}$ are called NPN-equivalent.
$\Longrightarrow$ given a penalty function for F, computing one for F^* is trivial.

$x_3 \leftrightarrow (x_1 \wedge x_2)$, $\neg x_3 \leftrightarrow (\neg x_1 \wedge \neg x_2)$, $x_3 \leftrightarrow (x_1 \vee x_2)$, $x_3 \leftrightarrow (x_1 \rightarrow x_2)$, ... are NPN-equivalent

Properties of penalty functions: NPN-Equivalence

Let $F(\underline{\mathbf{x}})$ be a Boolean function, and let $F^*(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} F(x_1, ..., x_{i-1}, \neg x_i, x_{i+1}, ..., x_n)$ for some index i.
Suppose also that $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ is a penalty function for $F(\underline{\mathbf{x}})$ under some placement.
Then $P_{F^*}(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}^*)$, where $\underline{\theta}^*$ is s.t., for every $z, z' \in \underline{\mathbf{x}}, \underline{\mathbf{a}}$:

$$\theta^*_z = \begin{cases} -\theta_z & \textit{if } z = x_i \\ \theta_z & \textit{otherwise} \end{cases} \qquad \theta^*_{zz'} = \begin{cases} -\theta_{zz'} & \textit{if } z = x_i \textit{ or } z' = x_i \\ \theta_{zz'} & \textit{otherwise} \end{cases}$$

Example

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\ \ x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\neg x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} + \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 - \frac{1}{2}x_1x_2 + x_1x_3 - x_2a - x_3a$

Two Boolean functions $F(\underline{\mathbf{x}}), F^*(\underline{\mathbf{x}})$ that become equivalent by permuting or negating some of their input variables $\underline{\mathbf{x}}$ are called NPN-equivalent.
$\Longrightarrow$ given a penalty function for F, computing one for F^* is trivial.

$x_3 \leftrightarrow (x_1 \wedge x_2)$, $\neg x_3 \leftrightarrow (\neg x_1 \wedge \neg x_2)$, $x_3 \leftrightarrow (x_1 \vee x_2)$, $x_3 \leftrightarrow (x_1 \rightarrow x_2)$, ... are NPN-equivalent

Properties of penalty functions: NPN-Equivalence

Let $F(\underline{\mathbf{x}})$ be a Boolean function, and let $F^*(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} F(x_1, ..., x_{i-1}, \neg x_i, x_{i+1}, ..., x_n)$ for some index i.
Suppose also that $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ is a penalty function for $F(\underline{\mathbf{x}})$ under some placement.
Then $P_{F^*}(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}^*)$, where $\underline{\theta}^*$ is s.t., for every $z, z' \in \underline{\mathbf{x}}, \underline{\mathbf{a}}$:

$$\theta^*_z = \begin{cases} -\theta_z & \text{if } z = x_i \\ \theta_z & \text{otherwise} \end{cases} \qquad \theta^*_{zz'} = \begin{cases} -\theta_{zz'} & \text{if } z = x_i \text{ or } z' = x_i \\ \theta_{zz'} & \text{otherwise} \end{cases}$$

Example

if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\ \ x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 + \frac{1}{2}x_1x_2 - x_1x_3 - x_2a - x_3a$
if $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_3 \leftrightarrow (\neg x_1 \wedge x_2)$, then $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = \frac{5}{2} + \frac{1}{2}x_1 - \frac{1}{2}x_2 + x_3 - \frac{1}{2}x_1x_2 + x_1x_3 - x_2a - x_3a$

Two Boolean functions $F(\underline{\mathbf{x}}), F^*(\underline{\mathbf{x}})$ that become equivalent by permuting or negating some of their input variables $\underline{\mathbf{x}}$ are called NPN-equivalent.
$\Longrightarrow$ given a penalty function for F, computing one for F^* is trivial.

$x_3 \leftrightarrow (x_1 \wedge x_2)$, $\neg x_3 \leftrightarrow (\neg x_1 \wedge \neg x_2)$, $x_3 \leftrightarrow (x_1 \vee x_2)$, $x_3 \leftrightarrow (x_1 \rightarrow x_2)$, ... are NPN-equivalent

Properties of penalty functions: And-Decomposition

Given $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^k)$, each F_k with p.f. $P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k)$ and gap g^k_{min}
s.t. $\underline{\mathbf{x}} = \cup_k \underline{\mathbf{x}}^k$, $\underline{\mathbf{a}} = \cup_k \underline{\mathbf{a}}^k$, the $\underline{\mathbf{a}}^k$s all disjoint, the $\underline{\mathbf{x}}^k$s not disjoint ;
if $\theta_i = \sum_k \theta_i^k \quad \in [-2, 2]$ and $\theta_{ij} = \sum_k \theta_{ij}^k \quad \in [-1, 1]$.
$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k)$ p.f. for $F(\underline{\mathbf{x}})$, $g_{min} = \min_k(g^k_{min} \quad)$.

If $\sum_k \theta_i^k$ and $\sum_k \theta_{ij}$ on shared x_is x_js violate the ranges
weights w_k may be introduced $\Longrightarrow$ the gap is weakened

When two conjuncts F_k, $F_{k'}$ share one variable x_i,
rename the second occurrence with a fresh one x_i', conjoining $(x_i \leftrightarrow x_i')$:
$F^*(\underline{\mathbf{x}}^*) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^{k*}) \wedge \bigwedge_{\{x_i \ shared\}} (x_i \leftrightarrow x_i')$
$\Longrightarrow P_{F^*}(\underline{\mathbf{x}}^*, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k) + \sum_{\{x_i \ shared\}} (1 - x_i x_i')$
$\Longrightarrow g_{min} = \min_k(g^k_{min}, 2)$.

$(x_i \leftrightarrow x_i')$ produces $1 - x_i x_i'$ of gap 2 $\Longrightarrow$ no parameter range issue

Properties of penalty functions: And-Decomposition

Given $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^k)$, each F_k with p.f. $P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k)$ and gap g^k_{min}
s.t. $\underline{\mathbf{x}} = \cup_k \underline{\mathbf{x}}^k$, $\underline{\mathbf{a}} = \cup_k \underline{\mathbf{a}}^k$, the $\underline{\mathbf{a}}^k$s all disjoint, the $\underline{\mathbf{x}}^k$s not disjoint ;
given $w_k > 0$ s.t. $\theta_i = \sum_k \theta_i^k w_k \in [-2, 2]$ and $\theta_{ij} = \sum_k \theta_{ij}^k w_k \in [-1, 1]$.
$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k) w_k$ p.f. for $F(\underline{\mathbf{x}})$, $g_{min} = \min_k(g^k_{min} w_k)$.

If $\sum_k \theta_i^k$ and $\sum_k \theta_{ij}$ on shared x_is x_js violate the ranges
weights w_k may be introduced $\Longrightarrow$ the gap is weakened

When two conjuncts F_k, $F_{k'}$ share one variable x_i,
rename the second occurrence with a fresh one x_i', conjoining $(x_i \leftrightarrow x_i')$:
$F^*(\underline{\mathbf{x}}^*) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^{k*}) \wedge \bigwedge_{\{x_i \ shared\}} (x_i \leftrightarrow x_i')$
$\Longrightarrow P_{F^*}(\underline{\mathbf{x}}^*, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k|\underline{\boldsymbol{\theta}}^k) + \sum_{\{x_i \ shared\}} (1 - x_i x_i')$
$\Longrightarrow g_{min} = \min_k(g^k_{min}, 2)$.

$(x_i \leftrightarrow x_i')$ produces $1 - x_i x_i'$ of gap 2 $\Longrightarrow$ no parameter range issue

Properties of penalty functions: And-Decomposition

Given $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^k)$, each F_k with p.f. $P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k | \underline{\boldsymbol{\theta}}^k)$ and gap g^k_{min}
s.t. $\underline{\mathbf{x}} = \cup_k \underline{\mathbf{x}}^k$, $\underline{\mathbf{a}} = \cup_k \underline{\mathbf{a}}^k$, the $\underline{\mathbf{a}}^k$s all disjoint, the $\underline{\mathbf{x}}^k$s not disjoint ;
if $\theta_i = \sum_k \theta^k_i \quad \in [-2, 2]$ and $\theta_{ij} = \sum_k \theta^k_{ij} \quad \in [-1, 1]$.
$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k | \underline{\boldsymbol{\theta}}^k)$ p.f. for $F(\underline{\mathbf{x}})$, $g_{min} = \min_k(g^k_{min} \quad)$.

If $\sum_k \theta^k_i$ and $\sum_k \theta_{ij}$ on shared x_is x_js violate the ranges
weights w_k may be introduced $\Longrightarrow$ the gap is weakened

When two conjuncts F_k, $F_{k'}$ share one variable x_i,
rename the second occurrence with a fresh one x'_i, conjoining $(x_i \leftrightarrow x'_i)$:
$F^*(\underline{\mathbf{x}}^*) \stackrel{\text{def}}{=} \bigwedge_k F_k(\underline{\mathbf{x}}^{k*}) \wedge \bigwedge_{\{x_i \ shared\}} (x_i \leftrightarrow x'_i)$
$\Longrightarrow P_{F^*}(\underline{\mathbf{x}}^*, \underline{\mathbf{a}} | \underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sum_k P_{F_k}(\underline{\mathbf{x}}^k, \underline{\mathbf{a}}^k | \underline{\boldsymbol{\theta}}^k) + \sum_{\{x_i \ shared\}} (1 - x_i x'_i)$
$\Longrightarrow g_{min} = \min_k(g^k_{min}, 2)$.

$(x_i \leftrightarrow x'_i)$ produces $1 - x_i x'_i$ of gap 2 $\Longrightarrow$ no parameter range issue

Formula decomposition

1. Tseitin-style decompose $F(\underline{\mathbf{x}})$ into an equi-satisfiable formula:
   $F^*(\underline{\mathbf{x}}, \underline{\mathbf{y}}) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{m-1}(y_i \leftrightarrow F_i(\underline{\mathbf{x}}^i, \underline{\mathbf{y}}^i)) \wedge F_m(\underline{\mathbf{x}}^m, \underline{\mathbf{y}}^m) \qquad (y_i \text{ fresh})$
2. when two conjuncts F_1, F_2 share one variable y_j, rename the second with a fresh one y_i', conjoining $(y_i \leftrightarrow y_i')$
3. compute separately the penalty functions of the conjuncts
4. sum them

Formula decomposition: example

- $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_4 \leftrightarrow (x_3 \wedge (x_1 \oplus x_2))$

$\Longrightarrow F^*(\underline{\mathbf{x}}, y) \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y \leftrightarrow (x_1 \oplus x_2))$

$\Longrightarrow F^{**}(\underline{\mathbf{x}}, y, y') \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y' \leftrightarrow (x_1 \oplus x_2)) \wedge (y \leftrightarrow y')$

$$\Longrightarrow P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \left(\begin{array}{l} \frac{5}{2} - \frac{1}{2}x_3 - \frac{1}{2}y' + x_4 + \frac{1}{2}x_3 y' - x_3 x_4 - y' a_4 - x_4 a_4 + \\ 5 + y + a_2 - a_3 + x_1 a_1 - x_1 a_2 - x_1 a_3 - x_2 a_1 - x_2 a_2 - x_2 a_3 + y a_2 - y a_3 \\ +1 - yy' \end{array} \right)$$

Formula decomposition: example

- $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_4 \leftrightarrow (x_3 \wedge (x_1 \oplus x_2))$

$\Longrightarrow F^*(\underline{\mathbf{x}}, y) \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y \leftrightarrow (x_1 \oplus x_2))$

$\Longrightarrow F^{**}(\underline{\mathbf{x}}, y, y') \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y' \leftrightarrow (x_1 \oplus x_2)) \wedge (y \leftrightarrow y')$

$$\Longrightarrow P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \begin{pmatrix} \frac{5}{2} - \frac{1}{2}x_3 - \frac{1}{2}y' + x_4 + \frac{1}{2}x_3 y' - x_3 x_4 - y' a_4 - x_4 a_4 + \\ 5 + y + a_2 - a_3 + x_1 a_1 - x_1 a_2 - x_1 a_3 - x_2 a_1 - x_2 a_2 - x_2 a_3 + y a_2 - y a_3 \\ +1 - y y' \end{pmatrix}$$

Formula decomposition: example

- $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_4 \leftrightarrow (x_3 \wedge (x_1 \oplus x_2))$

$\Longrightarrow F^*(\underline{\mathbf{x}}, y) \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y \leftrightarrow (x_1 \oplus x_2))$

$\Longrightarrow F^{**}(\underline{\mathbf{x}}, y, y') \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y' \leftrightarrow (x_1 \oplus x_2)) \wedge (y \leftrightarrow y')$

$$\Longrightarrow P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \begin{pmatrix} \frac{5}{2} - \frac{1}{2}x_3 - \frac{1}{2}y' + x_4 + \frac{1}{2}x_3 y' - x_3 x_4 - y' a_4 - x_4 a_4 + \\ 5 + y + a_2 - a_3 + x_1 a_1 - x_1 a_2 - x_1 a_3 - x_2 a_1 - x_2 a_2 - x_2 a_3 + y a_2 - y a_3 \\ +1 - yy' \end{pmatrix}$$

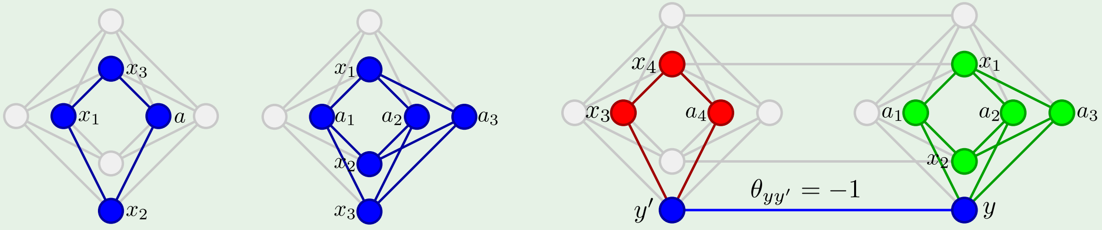
Formula decomposition: example

- $F(\underline{\mathbf{x}}) \stackrel{\text{def}}{=} x_4 \leftrightarrow (x_3 \wedge (x_1 \oplus x_2))$

$\Longrightarrow F^*(\underline{\mathbf{x}}, y) \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y \leftrightarrow (x_1 \oplus x_2))$

$\Longrightarrow F^{**}(\underline{\mathbf{x}}, y, y') \stackrel{\text{def}}{=} (x_4 \leftrightarrow (x_3 \wedge y)) \wedge (y' \leftrightarrow (x_1 \oplus x_2)) \wedge (y \leftrightarrow y')$

$$\Longrightarrow P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \underline{\theta}) = \begin{pmatrix} \frac{5}{2} - \frac{1}{2}x_3 - \frac{1}{2}y' + x_4 + \frac{1}{2}x_3 y' - x_3 x_4 - y' a_4 - x_4 a_4 + \\ 5 + y + a_2 - a_3 + x_1 a_1 - x_1 a_2 - x_1 a_3 - x_2 a_1 - x_2 a_2 - x_2 a_3 + y a_2 - y a_3 \\ +1 - yy' \end{pmatrix}$$



$x_3 \leftrightarrow (x_1 \wedge x_2)$ $\quad$ $x_3 \leftrightarrow (x_1 \oplus x_2)$ $\quad$ $x_4 \leftrightarrow (x_3 \wedge (x_1 \oplus x_2))$

Outline

1. Background: D-Wave's Quantum Annealers
2. Motivations & Goals
3. Solving SAT with Quantum Annealers: Foundations
   - Penalty Functions
   - Encoding SAT into Penalty Functions
   - Some Issues
4. Preliminary Empirical Results

Encoding (small) sub-formulas into penalty functions

General "Monolithic" Encoding Problem

Assume some bit-to-qubit map $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$ (placement) and let:

$$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \theta_0 + \overbrace{\sum \theta_i x_i + \sum \theta_{ij} x_i x_j}^{\text{part without ancillas}} + \overbrace{\sum \theta_i a_i + \sum \theta_{ij} a_i x_j + \sum \theta_{ij} a_i a_j}^{\text{part with ancillas}}$$

$$\text{find } \underline{\theta} \text{ s.t. } \forall \underline{\mathbf{x}}. \left[\begin{array}{l} (\quad F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq 0)) \wedge \\ (\quad F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 0)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq g_{min})) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = g_{min})) \end{array} \right]$$

$$\text{and s.t. } \theta_i \in [-2, 2], \theta_{ij} \in [-1, 1], \forall i, j$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ to be exact

- Quantifiers can be Shannon-expanded into a SMT($\mathcal{LRA}$) formula (SMT-based encoding)

Encoding (small) sub-formulas into penalty functions

General "Monolithic" Encoding Problem

Assume some bit-to-qubit map $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$ (placement) and let:

$$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \stackrel{\text{def}}{=} \theta_0 + \overbrace{\sum \theta_i x_i + \sum \theta_{ij} x_i x_j}^{\text{part without ancillas}} + \overbrace{\sum \theta_i a_i + \sum \theta_{ij} a_i x_j + \sum \theta_{ij} a_i a_j}^{\text{part with ancillas}}$$

$$\textit{find } \underline{\theta} \textit{ s.t. } \forall \underline{\mathbf{x}}. \left[\begin{array}{l} (\ \ F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq 0)) \wedge \\ (\ \ F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 0)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq g_{min})) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = g_{min})) \end{array} \right]$$

$$\textit{and s.t. } \theta_i \in [-2, 2], \theta_{ij} \in [-1, 1], \ \forall i, j$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ to be exact

- Quantifiers can be Shannon-expanded into a SMT($\mathcal{LRA}$) formula (SMT-based encoding)

Encoding (small) sub-formulas into penalty functions

General "Monolithic" Encoding Problem

Assume some bit-to-qubit map $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$ (placement) and let:

$$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \stackrel{\text{def}}{=} \theta_0 + \overbrace{\sum \theta_i x_i + \sum \theta_{ij} x_i x_j}^{\text{part without ancillas}} + \overbrace{\sum \theta_i a_i + \sum \theta_{ij} a_i x_j + \sum \theta_{ij} a_i a_j}^{\text{part with ancillas}}$$

$$\text{find } \underline{\theta} \text{ s.t. } \forall \underline{\mathbf{x}}. \left[\begin{array}{l} (\quad F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq 0)) \wedge \\ (\quad F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 0)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq g_{min})) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = g_{min})) \end{array} \right]$$

$$\text{and s.t. } \theta_i \in [-2, 2], \theta_{ij} \in [-1, 1], \ \forall i, j$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ to be exact

- Quantifiers can be Shannon-expanded into a SMT($\mathcal{LRA}$) formula (SMT-based encoding)

Encoding (small) sub-formulas into penalty functions

General "Monolithic" Encoding Problem

Assume some bit-to-qubit map $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$ (placement) and let:

$$P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \stackrel{\text{def}}{=} \theta_0 + \overbrace{\sum \theta_i x_i + \sum \theta_{ij} x_i x_j}^{\text{part without ancillas}} + \overbrace{\sum \theta_i a_i + \sum \theta_{ij} a_i x_j + \sum \theta_{ij} a_i a_j}^{\text{part with ancillas}}$$

$$\textit{find } \underline{\theta} \textit{ s.t. } \forall \underline{\mathbf{x}}. \left[\begin{array}{l} (\quad F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq 0)) \wedge \\ (\quad F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 0)) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \forall \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \geq g_{min})) \wedge \\ (\neg F(\underline{\mathbf{x}}) \rightarrow \exists \underline{\mathbf{a}}.(P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = g_{min})) \end{array} \right]$$

$$\textit{and s.t. } \theta_i \in [-2, 2], \theta_{ij} \in [-1, 1], \ \forall i, j$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ to be exact

- Quantifiers can be Shannon-expanded into a SMT($\mathcal{LRA}$) formula (SMT-based encoding)

Encoding (small) sub-formulas into penalty functions

- Given a placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$, finding $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}})$ (i.e., the θ's) reduces to the SMT($\mathcal{LRA}$) problem:

$$
\begin{aligned}
\Phi(\underline{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} & \bigwedge_{z_i \in \underline{\mathbf{x}}, \underline{\mathbf{a}}} (-2 \le \theta_i) \wedge (\theta_i \le 2) \\
\wedge & \bigwedge_{z_i, z_j \in \underline{\mathbf{x}}, \underline{\mathbf{a}},\ i<j} (-1 \le \theta_{ij}) \wedge (\theta_{ij} \le 1) \\
\wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \top\}} \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \ge 0) \\
\wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \top\}} \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) = 0) \\
\wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \bot\}} \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) \ge g_{min}) \\
\wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \bot\}} \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}}) = g_{min}).
\end{aligned}
$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}})$ to be exact

$\Longrightarrow$ Finding a maximum-gap $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\boldsymbol{\theta}})$ reduces to a problem in optimization modulo theories (OMT($\mathcal{LRA}$)): $\langle \Phi(\underline{\boldsymbol{\theta}}), g_{min} \rangle$

Encoding (small) sub-formulas into penalty functions

- Given a placement $\underline{\mathbf{x}}, \underline{\mathbf{a}} \longmapsto \underline{\mathbf{z}}$, finding $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ (i.e., the θ's) reduces to the SMT($\mathcal{LRA}$) problem:

$$
\begin{array}{lcl}
\Phi(\underline{\theta}) & \stackrel{\text{def}}{=} & \bigwedge_{z_i \in \underline{\mathbf{x}}, \underline{\mathbf{a}}} (-2 \le \theta_i) \wedge (\theta_i \le 2) \\
& \wedge & \bigwedge_{z_i, z_j \in \underline{\mathbf{x}}, \underline{\mathbf{a}},\ i<j} (-1 \le \theta_{ij}) \wedge (\theta_{ij} \le 1) \\
& \wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \top\}} \ \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \ge 0) \\
& \wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \top\}} \ \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = 0) \\
& \wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \bot\}} \ \bigwedge_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) \ge g_{min}) \\
& \wedge & \bigwedge_{\{\underline{\mathbf{x}} \in \{-1,1\}^n | F(\underline{\mathbf{x}}) = \bot\}} \ \bigvee_{\underline{\mathbf{a}} \in \{-1,1\}^h} (P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta}) = g_{min}).
\end{array}
$$

if we want $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ to be exact

$\Longrightarrow$ Finding a maximum-gap $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}}|\underline{\theta})$ reduces to a problem in optimization modulo theories (OMT($\mathcal{LRA}$)): $\langle \Phi(\underline{\theta}), g_{min} \rangle$

Combined placement and encoding

- Let $\underline{\mathbf{z}} \stackrel{\text{def}}{=} \{x_1, ..., x_n, a_1, ..., a_h\}$, $V \stackrel{\text{def}}{=} \{1, ..., n+h\}$ the target qubits
- Introduce a set of $n+h$ *integer* variables $\underline{\mathbf{v}} \stackrel{\text{def}}{=} \{v_1, ..., v_{n+h}\}$
  - v_j: the vertex into which z_j is placed.
  - add $Distinct(v_1, ..., v_{n+h}) \wedge \bigwedge_{1 \le j \le n+h} (1 \le v_j) \wedge (v_j \le n+h)$
- Introduce $\mathrm{b} : V \longmapsto \mathbb{Q}$ ("bias") and $\mathrm{c} : V \times V \longmapsto \mathbb{Q}$ ("coupling"), rewrite $\Phi(\underline{\theta})$ into the $\mathcal{LRIA} \cup \mathcal{UF}$ formula $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$:
  - in $\Phi(\underline{\theta})$, replace θ_j and θ_{ij} by $\mathrm{b}(v_j)$ and $\mathrm{c}(v_i, v_j)$:
    $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}}) = \theta_0 + \sum_j \mathrm{b}(v_j) \cdot z_j + \sum_{i,j} \mathrm{c}(v_i, v_j) \cdot z_i \cdot z_j$
  - add connectivity graph description (plus few other constraints) :
    $\bigwedge_{\langle i,j \rangle \notin E} (\mathrm{c}(i,j) = 0)$
- Feed $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$ to a SMT/OMT($\mathcal{LRIA} \cup \mathcal{UF}$) solver
  $\Longrightarrow$ solution provides both the placement $\underline{\mathbf{v}}$ and the values for offset θ_0, biases $\mathrm{b}(v_j)$ and couplings $\mathrm{c}(v_i, v_j)$
- Placement improved by adding symmetry-breaking constraints (e.g., from $8! = 40320$ to $\binom{7}{3} = 35$ candidate placements in 1 tile)

Combined placement and encoding

- Let $\underline{\mathbf{z}} \stackrel{\text{def}}{=} \{x_1, ..., x_n, a_1, ..., a_h\}$, $V \stackrel{\text{def}}{=} \{1, ..., n+h\}$ the target qubits
- Introduce a set of $n+h$ *integer* variables $\underline{\mathbf{v}} \stackrel{\text{def}}{=} \{v_1, ..., v_{n+h}\}$
  - v_j: the vertex into which z_j is placed.
  - add $\mathit{Distinct}(v_1, ..., v_{n+h}) \wedge \bigwedge_{1 \le j \le n+h} (1 \le v_j) \wedge (v_j \le n+h)$
- Introduce $\mathsf{b} : V \longmapsto \mathbb{Q}$ ("bias") and $\mathsf{c} : V \times V \longmapsto \mathbb{Q}$ ("coupling"), rewrite $\Phi(\underline{\theta})$ into the $\mathcal{LRIA} \cup \mathcal{UF}$ formula $\Phi(\theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}})$:
  - in $\Phi(\underline{\theta})$, replace θ_j and θ_{ij} by $\mathsf{b}(v_j)$ and $\mathsf{c}(v_i, v_j)$:
    $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}}) = \theta_0 + \sum_j \mathsf{b}(v_j) \cdot z_j + \sum_{i,j} \mathsf{c}(v_i, v_j) \cdot z_i \cdot z_j$
  - add connectivity graph description (plus few other constraints) :
    $\bigwedge_{\langle i,j \rangle \notin E} (\mathsf{c}(i,j) = 0)$
- Feed $\Phi(\theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}})$ to a SMT/OMT($\mathcal{LRIA} \cup \mathcal{UF}$) solver
  $\Longrightarrow$ solution provides both the placement $\underline{\mathbf{v}}$ and the values for offset θ_0, biases $\mathsf{b}(v_j)$ and couplings $\mathsf{c}(v_i, v_j)$
- Placement improved by adding symmetry-breaking constraints
  (e.g., from $8! = 40320$ to $\binom{7}{3} = 35$ candidate placements in 1 tile)

Combined placement and encoding

- Let $\underline{\mathbf{z}} \stackrel{\text{def}}{=} \{x_1, ..., x_n, a_1, ..., a_h\}$, $V \stackrel{\text{def}}{=} \{1, ..., n+h\}$ the target qubits
- Introduce a set of $n+h$ *integer* variables $\underline{\mathbf{v}} \stackrel{\text{def}}{=} \{v_1, ..., v_{n+h}\}$
  - v_j: the vertex into which z_j is placed.
  - add $\mathit{Distinct}(v_1, ..., v_{n+h}) \wedge \bigwedge_{1 \le j \le n+h} (1 \le v_j) \wedge (v_j \le n+h)$
- Introduce $\mathrm{b} : V \longmapsto \mathbb{Q}$ ("bias") and $\mathrm{c} : V \times V \longmapsto \mathbb{Q}$ ("coupling"), rewrite $\Phi(\underline{\boldsymbol{\theta}})$ into the $\mathcal{LRIA} \cup \mathcal{UF}$ formula $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$:
  - in $\Phi(\underline{\boldsymbol{\theta}})$, replace θ_j and θ_{ij} by $\mathrm{b}(v_j)$ and $\mathrm{c}(v_i, v_j)$:
    $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}}) = \theta_0 + \sum_j \mathrm{b}(v_j) \cdot z_j + \sum_{i,j} \mathrm{c}(v_i, v_j) \cdot z_i \cdot z_j$
  - add connectivity graph description (plus few other constraints) :
    $\bigwedge_{\langle i,j \rangle \notin E} (\mathrm{c}(i,j) = 0)$
- Feed $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$ to a SMT/OMT($\mathcal{LRIA} \cup \mathcal{UF}$) solver
  $\Longrightarrow$ solution provides both the placement $\underline{\mathbf{v}}$ and the values for offset θ_0, biases $\mathrm{b}(v_j)$ and couplings $\mathrm{c}(v_i, v_j)$
- Placement improved by adding symmetry-breaking constraints (e.g., from $8! = 40320$ to $\binom{7}{3} = 35$ candidate placements in 1 tile)

Combined placement and encoding

- Let $\underline{\mathbf{z}} \stackrel{\text{def}}{=} \{x_1, ..., x_n, a_1, ..., a_h\}$, $V \stackrel{\text{def}}{=} \{1, ..., n+h\}$ the target qubits
- Introduce a set of $n+h$ *integer* variables $\underline{\mathbf{v}} \stackrel{\text{def}}{=} \{v_1, ..., v_{n+h}\}$
  - v_j: the vertex into which z_j is placed.
  - add $\mathit{Distinct}(v_1, ..., v_{n+h}) \wedge \bigwedge_{1 \le j \le n+h} (1 \le v_j) \wedge (v_j \le n+h)$
- Introduce $\mathrm{b} : V \longmapsto \mathbb{Q}$ ("bias") and $\mathrm{c} : V \times V \longmapsto \mathbb{Q}$ ("coupling"), rewrite $\Phi(\underline{\boldsymbol{\theta}})$ into the $\mathcal{LRIA} \cup \mathcal{UF}$ formula $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$:
  - in $\Phi(\underline{\boldsymbol{\theta}})$, replace θ_j and θ_{ij} by $\mathrm{b}(v_j)$ and $\mathrm{c}(v_i, v_j)$:
    $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}}) = \theta_0 + \sum_j \mathrm{b}(v_j) \cdot z_j + \sum_{i,j} \mathrm{c}(v_i, v_j) \cdot z_i \cdot z_j$
  - add connectivity graph description (plus few other constraints) :
    $\bigwedge_{\langle i,j \rangle \notin E} (\mathrm{c}(i, j) = 0)$
- Feed $\Phi(\theta_0, \mathrm{b}, \mathrm{c}, \underline{\mathbf{v}})$ to a SMT/OMT($\mathcal{LRIA} \cup \mathcal{UF}$) solver
  $\Longrightarrow$ solution provides both the placement $\underline{\mathbf{v}}$ and the values for offset θ_0, biases $\mathrm{b}(v_j)$ and couplings $\mathrm{c}(v_i, v_j)$
- Placement improved by adding symmetry-breaking constraints (e.g., from $8! = 40320$ to $\binom{7}{3} = 35$ candidate placements in 1 tile)

Combined placement and encoding

- Let $\underline{\mathbf{z}} \stackrel{\text{def}}{=} \{x_1, ..., x_n, a_1, ..., a_h\}$, $V \stackrel{\text{def}}{=} \{1, ..., n+h\}$ the target qubits
- Introduce a set of $n+h$ *integer* variables $\underline{\mathbf{v}} \stackrel{\text{def}}{=} \{v_1, ..., v_{n+h}\}$
  - v_j: the vertex into which z_j is placed.
  - add $\mathit{Distinct}(v_1, ..., v_{n+h}) \wedge \bigwedge_{1 \le j \le n+h} (1 \le v_j) \wedge (v_j \le n+h)$
- Introduce $\mathsf{b} : V \longmapsto \mathbb{Q}$ ("bias") and $\mathsf{c} : V \times V \longmapsto \mathbb{Q}$ ("coupling"), rewrite $\Phi(\underline{\boldsymbol{\theta}})$ into the $\mathcal{LRIA} \cup \mathcal{UF}$ formula $\Phi(\theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}})$:
  - in $\Phi(\underline{\boldsymbol{\theta}})$, replace θ_j and θ_{ij} by $\mathsf{b}(v_j)$ and $\mathsf{c}(v_i, v_j)$:
    $P_F(\underline{\mathbf{x}}, \underline{\mathbf{a}} | \theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}}) = \theta_0 + \sum_j \mathsf{b}(v_j) \cdot z_j + \sum_{i,j} \mathsf{c}(v_i, v_j) \cdot z_i \cdot z_j$
  - add connectivity graph description (plus few other constraints) :
    $\bigwedge_{\langle i,j \rangle \notin E} (\mathsf{c}(i,j) = 0)$
- Feed $\Phi(\theta_0, \mathsf{b}, \mathsf{c}, \underline{\mathbf{v}})$ to a SMT/OMT($\mathcal{LRIA} \cup \mathcal{UF}$) solver
  $\Longrightarrow$ solution provides both the placement $\underline{\mathbf{v}}$ and the values for offset θ_0, biases $\mathsf{b}(v_j)$ and couplings $\mathsf{c}(v_i, v_j)$
- Placement improved by adding symmetry-breaking constraints
  (e.g., from $8! = 40320$ to $\binom{7}{3} = 35$ candidate placements in 1 tile)

SMT/OMT-based "monolithic" encoding: limitations

Complexity/size

- General formula is $\exists\forall\exists$-quantified $\Longrightarrow$ solving worse than NP!
- SMT-based encoding $\Phi(\underline{\theta})$ exponential wrt. $|\underline{\mathbf{x}}, \underline{\mathbf{a}}|$
- Improvements: symmetry break, variable elimination (see paper)

Over-constrainedness

- The monolithic encoding problem is very over-constrained:
  - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels ($O(2^{|\underline{\mathbf{x}}|})$ equalities + $O(2^{|\underline{\mathbf{x}}|+|\underline{\mathbf{a}}|})$ inequalities)
  - $O(|\underline{\mathbf{x}}| + |\underline{\mathbf{a}}|)$ degrees of freedom: θ_i, θ_{ij}

$\Longrightarrow$ to have a satisfiable $\Phi(\underline{\theta})$, the number of ancillas $|\underline{\mathbf{a}}|$ grows

$\Longrightarrow$ large waste of qubits

$\Longrightarrow$ works only for small input Boolean formulas $F(\underline{\mathbf{x}})$

$\Longrightarrow$ medium/large formulas need decomposing

SMT/OMT-based "monolithic" encoding: limitations

Complexity/size

- General formula is $\exists\forall\exists$-quantified $\Longrightarrow$ solving worse than NP!
- SMT-based encoding $\Phi(\underline{\boldsymbol{\theta}})$ exponential wrt. $|\underline{\mathbf{x}}, \underline{\mathbf{a}}|$
- Improvements: symmetry break, variable elimination (see paper)

Over-constrainedness

- The monolithic encoding problem is very over-constrained:
  - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels ($O(2^{|\underline{\mathbf{x}}|})$ equalities + $O(2^{|\underline{\mathbf{x}}|+|\underline{\mathbf{a}}|})$ inequalities)
  - $O(|\underline{\mathbf{x}}| + |\underline{\mathbf{a}}|)$ degrees of freedom: θ_i, θ_{ij}

$\Longrightarrow$ to have a satisfiable $\Phi(\underline{\boldsymbol{\theta}})$, the number of ancillas $|\underline{\mathbf{a}}|$ grows
$\Longrightarrow$ large waste of qubits

$\Longrightarrow$ works only for small input Boolean formulas $F(\underline{\mathbf{x}})$
$\Longrightarrow$ medium/large formulas need decomposing

SMT/OMT-based "monolithic" encoding: limitations

Complexity/size

- General formula is $\exists\forall\exists$-quantified $\Longrightarrow$ solving worse than NP!
- SMT-based encoding $\Phi(\underline{\theta})$ exponential wrt. $|\underline{\mathbf{x}}, \underline{\mathbf{a}}|$
- Improvements: symmetry break, variable elimination (see paper)

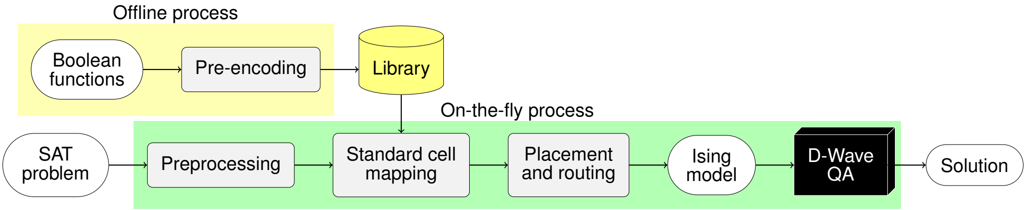
Over-constrainedness

- The monolithic encoding problem is very over-constrained:
  - must discriminate between $O(2^{|\underline{\mathbf{x}}|})$ models & countermodels ($O(2^{|\underline{\mathbf{x}}|})$ equalities + $O(2^{|\underline{\mathbf{x}}|+|\underline{\mathbf{a}}|})$ inequalities)
  - $O(|\underline{\mathbf{x}}| + |\underline{\mathbf{a}}|)$ degrees of freedom: θ_i, θ_{ij}

$\Longrightarrow$ to have a satisfiable $\Phi(\underline{\theta})$, the number of ancillas $|\underline{\mathbf{a}}|$ grows
$\Longrightarrow$ large waste of qubits

$\Longrightarrow$ works only for small input Boolean formulas $F(\underline{\mathbf{x}})$
$\Longrightarrow$ medium/large formulas need decomposing

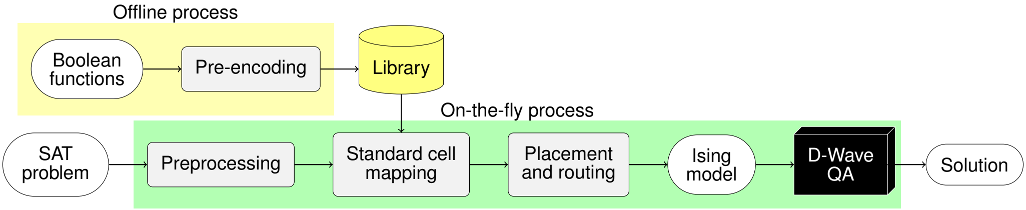
General schema



Offline process

- Pre-encoding: produce a library of penalty functions of small and useful Boolean sub-formulas (modulo NPN equivalence)
  - based on SMT/OMT
  - resource-intensive

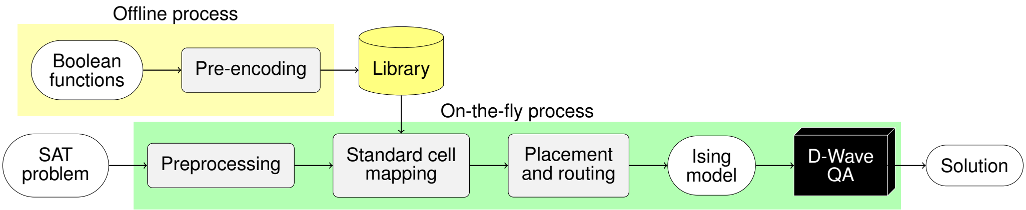
General schema



On-the-fly translation

- Preprocessing
  - Boolean formula simplifying & rewriting
  - relatively fast

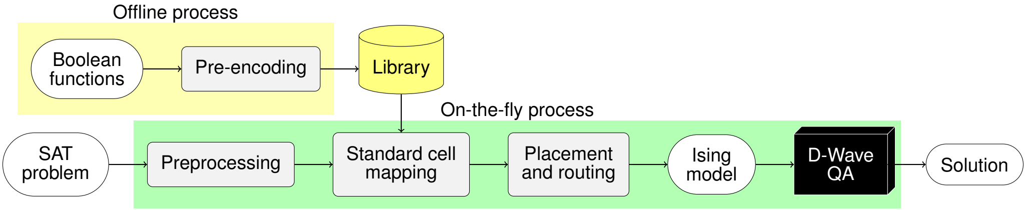
General schema



On-the-fly translation

- Cell mapping
  - decompose F into $\bigwedge_k F_k$ available from the library (modulo NPN-equivalence)
  - natural choice: F_ks typically fitting in 1 tile
  - efficient heuristics from logic synthesis for NPN-matching & technology mapping [Huang et al. 2013, Mishchenko et al 2005,07]

General schema



On-the-fly translation

- Placement & Routing
  - placement: each p.f. P_{F_k} is assigned a disjoint subgraph of the QA graph (typically 1 tile)
  - routing: equivalence chains of qubits (p.f. $1 - x_i x_i'$) built to connect variables shared by P_{F_k}s
  - efficient heuristics from design of digital circuits [Betz & Rose, 1997; Gester et al. 2013; ...]
  - currently our computational bottleneck
  - produces long chains: waste of qubits, make annealing less stable

Outline

Major problem for Q.A.: Equivalence Chains

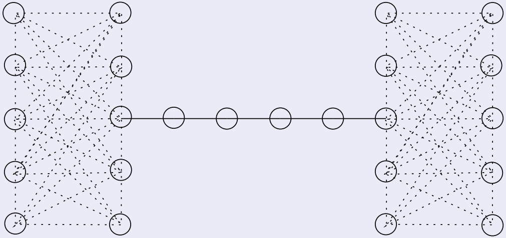
- Given $F(x,..) \wedge G(x,...)$, w. shared variable x, with p.f. P_F, P_G

$\Longrightarrow$ $F(x_0,..) \wedge G(x_n,...) \wedge \bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$

$\Longrightarrow$ $P_{F\wedge G}(...) = P_F(x_0,...) + P_G(x_n,...) + \sum_{i=1}^{n}(1 - x_{i-1}x_i)$

  - place $F(x_0,..)$, $G(x_n,...)$,
  - an equivalence chain $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$ links the two x instances

- Alternatively, if extended coupling range $[-2,1]$ applied, then
  $P_{F\wedge G}(...) = P_F(x_0,...) + P_G(x_n,...) + \sum_{i=1}^{n}(2 - 2x_{i-1}x_i)$
  $\Longrightarrow$ "stronger" chains (with gap 4)



Major problem for Q.A.: Equivalence Chains

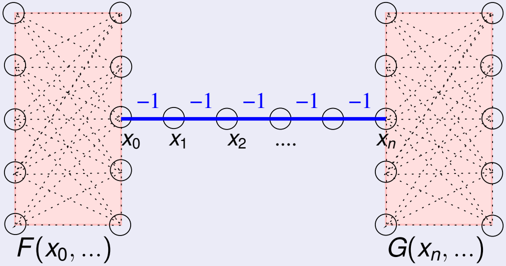
- Given $F(x, ..) \wedge G(x, ...)$, w. shared variable x, with p.f. P_F, P_G

$\Longrightarrow$ $F(x_0, ..) \wedge G(x_n, ...) \wedge \bigwedge_{i=1}^{n} (x_{i-1} \leftrightarrow x_i)$

$\Longrightarrow$ $P_{F \wedge G}(...) = P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n} (1 - x_{i-1} x_i)$

  - place $F(x_0, ..)$, $G(x_n, ...)$,
  - an equivalence chain $\bigwedge_{i=1}^{n} (x_{i-1} \leftrightarrow x_i)$ links the two x instances

- Alternatively, if extended coupling range $[-2, 1]$ applied, then $P_{F \wedge G}(...) = P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n} (2 - 2x_{i-1} x_i)$ $\Longrightarrow$ "stronger" chains (with gap 4)



Major problem for Q.A.: Equivalence Chains

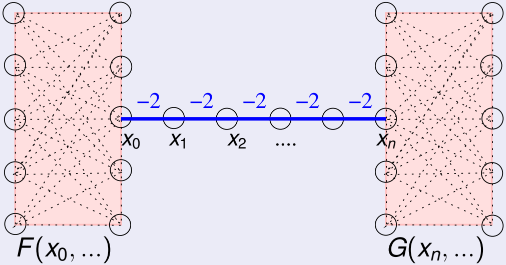
- Given $F(x, ..) \wedge G(x, ...)$, w. shared variable x, with p.f. P_F, P_G

$\Longrightarrow F(x_0, ..) \wedge G(x_n, ...) \wedge \bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$

$\Longrightarrow P_{F \wedge G}(...) = P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n}(1 - x_{i-1}x_i)$

  - place $F(x_0, ..)$, $G(x_n, ...)$,
  - an equivalence chain $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$ links the two x instances

- Alternatively, if extended coupling range $[-2, 1]$ applied, then
  $P_{F \wedge G}(...) = P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n}(2 - 2x_{i-1}x_i)$
  $\Longrightarrow$ "stronger" chains (with gap 4)



Major problem for Q.A.: Equivalence Chains

Problem: Equivalence chains "hard" for a Q.A. to deal with

- Unlike Boolean solvers, Q.A.s do not have equality propagation (BCP, ect.)
  - to "solve" $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$, must minimize $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ in
    $... + P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n}(1 - x_{i-1}x_i) + ...$
- Two typical failure scenarios:
  1. if gap of P_F's dominate over that of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$'s $[2 - 2x_ix_j]$, then Q.A. tends to "break" chains (e.g. $x_i = 1, x_{i+1} = -1$)
  2. if gap of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ $[2 - 2x_ix_j]$ dominate over that of P_F's, then Q.A. tends to "break" P_F's
- Problem: The bigger the qubits # in a chain ("chain strenght"), the harder for the Q.A. to toggle its value (without breaking it)
  - toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght n of the chain, not 1!
    $\Longrightarrow$ affects tunnelling
  - the Q.A. tends to assign values to stronger chains first, hard to modify eventually
    $\Longrightarrow$ it is desirable to have chains of homogeneous strength

Major problem for Q.A.: Equivalence Chains

Problem: Equivalence chains "hard" for a Q.A. to deal with

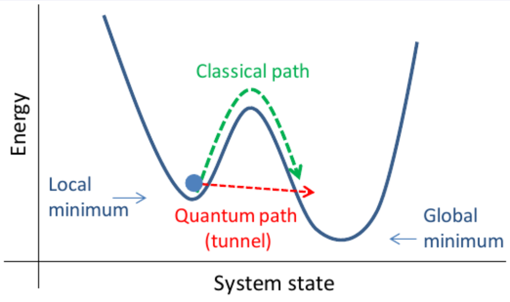
- Unlike Boolean solvers, Q.A.s do not have equality propagation (BCP, ect.)
  - to "solve" $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$, must minimize $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ in
    $... + P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n}(1 - x_{i-1}x_i) + ...$
- Two typical failure scenarios:
  1. if gap of P_F's dominate over that of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$'s $[2 - 2x_ix_j]$, then Q.A. tends to "break" chains (e.g. $x_i = 1, x_{i+1} = -1$)
  2. if gap of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ $[2 - 2x_ix_j]$ dominate over that of P_F's, then Q.A. tends to "break" P_F's
- Problem: The bigger the qubits # in a chain ("chain strenght"), the harder for the Q.A. to toggle its value (without breaking it)
  - toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght n of the chain, not 1!
    $\Longrightarrow$ affects tunnelling
  - the Q.A. tends to assign values to stronger chains first, hard to modify eventually
    $\Longrightarrow$ it is desirable to have chains of homogeneous strength

Major problem for Q.A.: Equivalence Chains

Problem: Equivalence chains “hard” for a Q.A. to deal with

- Unlike Boolean solvers, Q.A.s do not have equality propagation (BCP, ect.)
  - to “solve” $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$, must minimize $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ in $... + P_F(x_0, ...) + P_G(x_n, ...) + \sum_{i=1}^{n}(1 - x_{i-1}x_i) + ...$
- Two typical failure scenarios:
  1. if gap of P_F's dominate over that of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$'s $[2 - 2x_ix_j]$, then Q.A. tends to “break” chains (e.g. $x_i = 1, x_{i+1} = -1$)
  2. if gap of $\sum_{i=1}^{n}(1 - x_{i-1}x_i)$ $[2 - 2x_ix_j]$ dominate over that of P_F's, then Q.A. tends to “break” P_F's
- Problem: The bigger the qubits # in a chain (“chain strenght”), the harder for the Q.A. to toggle its value (without breaking it)
  - toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght n of the chain, not 1!
    $\Longrightarrow$ affects tunnelling
  - the Q.A. tends to assign values to stronger chains first, hard to modify eventually
    $\Longrightarrow$ it is desirable to have chains of homogeneous strength

Why equivalence chains affect quantum tunneling?

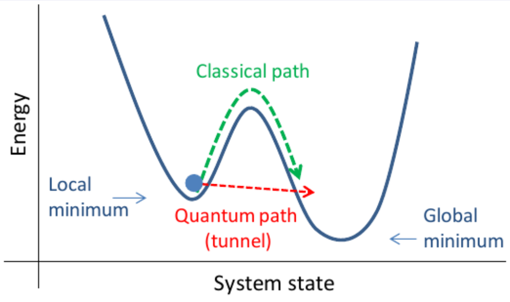


- (...)
- works well with tall & narrow energy barriers separating minima
- toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght of the chain

$\Longrightarrow$ strong chains may drastically enlarge the distance of minima

$\Longrightarrow$ reduce/hinder the effect of tunneling

Why equivalence chains affect quantum tunneling?

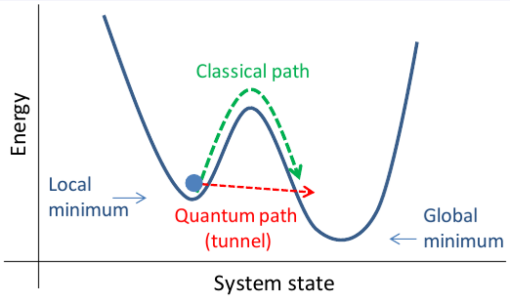


- (...)
- works well with tall & narrow energy barriers separating minima
- toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght of the chain

$\Longrightarrow$ strong chains may drastically enlarge the distance of minima

$\Longrightarrow$ reduce/hinder the effect of tunneling

Why equivalence chains affect quantum tunneling?



- (...)
- works well with tall & narrow energy barriers separating minima
- toggling a chain value moves from one state to another state whose Hamming dinstance is the strenght of the chain

$\Longrightarrow$ strong chains may drastically enlarge the distance of minima

$\Longrightarrow$ reduce/hinder the effect of tunneling

Remark

- Sometimes the Q.A. can break a chain $\bigwedge_{i=1}^{n}(x_{i-1} \leftrightarrow x_i)$ twice (or 2k times)

  $\Longrightarrow$ x_0 and x_n have the same value

  $\Longrightarrow$ correct solution, although energy $4k > 0$!
- Check the results for (unexpected) solutions!

Outline

Preliminary Empirical Results

Problem choice

- Used a variant of the sgen generator tool [Spence 2010]
  - generates very hard and small satisfiable Boolean formulas
  - parametric in the number of variables N
  - 100 sample formulas for each N
- Encoder uses OPTIMATHSAT OMT solver [Sebastiani & Trentin, 2015]

Remark

Distinct samples are statistically independent, so the probability $P_{\min}[N]$ of obtaining at least one minimum solution over N samples converges exponentially to 1 with N:

$$P_{\min}[N] = 1 - (1 - P_{\min}[1])^N.$$

D-Wave 2000Q assessment on SAT

D-Wave 2000Q (2048 qubits)		
Problem size	**# solved**	**% optimal samples**
32 vars	100	97.4
36 vars	100	96.4
40 vars	100	94.8
44 vars	100	93.8
48 vars	100	91.4
52 vars	100	93.4
56 vars	100	91.4
60 vars	100	88.2
64 vars	100	84.6
68 vars	100	84.4
72 vars	100	84.6
76 vars	100	86.6
80 vars	100	86.0

UBCSAT (SAPS)	
Problem size	**Avg time (ms)**
32 vars	0.1502
36 vars	0.2157
40 vars	0.3555
44 vars	0.5399
48 vars	0.8183
52 vars	1.1916
56 vars	1.4788
60 vars	2.2542
64 vars	3.1066
68 vars	4.8058
72 vars	6.2484
76 vars	8.2986
80 vars	12.4141

w. $>$80 vars could not fit in the QA graph

SAT problem instances

- 20 annealing samples per formula, 10μs each sample
  $\Longrightarrow$ 200μs total annealing time per formula

D-Wave 2000Q assessment on SAT

D-Wave 2000Q (2048 qubits)		
Problem size	**# solved**	**% optimal samples**
32 vars	100	97.4
36 vars	100	96.4
40 vars	100	94.8
44 vars	100	93.8
48 vars	100	91.4
52 vars	100	93.4
56 vars	100	91.4
60 vars	100	88.2
64 vars	100	84.6
68 vars	100	84.4
72 vars	100	84.6
76 vars	100	86.6
80 vars	100	86.0

UBCSAT (SAPS)	
Problem size	**Avg time (ms)**
32 vars	0.1502
36 vars	0.2157
40 vars	0.3555
44 vars	0.5399
48 vars	0.8183
52 vars	1.1916
56 vars	1.4788
60 vars	2.2542
64 vars	3.1066
68 vars	4.8058
72 vars	6.2484
76 vars	8.2986
80 vars	12.4141

w. $>$80 vars could not fit in the QA graph

w. $>$260 vars not solved by UBCSAT in 1000s

SAT problem instances

- s.o.a. SLS SAT solver UBCSAT with SAPS algorithm
  8-core Intel Xeon E5-2407 CPU, 2.20GHz.

D-Wave 2000Q assessment on SAT

D-Wave 2000Q (2048 qubits)		
Problem size	**# solved**	**% optimal samples**
32 vars	100	97.4
36 vars	100	96.4
40 vars	100	94.8
44 vars	100	93.8
48 vars	100	91.4
52 vars	100	93.4
56 vars	100	91.4
60 vars	100	88.2
64 vars	100	84.6
68 vars	100	84.4

UBCSAT (SAPS)	
Problem size	**Avg time (ms)**
32 vars	0.1502
36 vars	0.2157
40 vars	0.3555
44 vars	0.5399
48 vars	0.8183
52 vars	1.1916
56 vars	1.4788
60 vars	2.2542
64 vars	3.1066
68 vars	4.8058

Disclaimer

- Not a "comparison" wrt. UBCSAT / s.o.a. SAT solvers:
  - specialized HW vs. off-the-shelf HW
  - different timing mechanism & timing granularities
  - cost of encoding not considered here

$\Longrightarrow$ An empirical assessment of current Q.A. potentials