Pattern QUBOs

Pattern QUBOS: The Idea

$$\Phi = (x_1 \lor x_2 \lor \mathrm{x}_3) \land (x_1 \lor -x_2 \lor x_4) \land (x_1 \lor x_2 \lor x_4) \land (x_2 \lor -x_4 \lor x_3)$$

QUBO 1 **QUBO 2** **QUBO 3** **QUBO 4**

Combine

3-SAT QUBO

Pattern QUBOs: Clause types

Observation

All permutations of the variables within a clause are equal, i.e.:

$$(a \lor b \lor -\mathrm{c}) = (\mathrm{b} \ \lor -\mathrm{c} \lor a)$$

thus

Assume that **negated** variables are **always at the end** of the clause

Clause Types

Type 0: $(a \lor b \lor \mathrm{c})$

Type 1: $(a \lor b \lor -\mathrm{c})$

Type 2: $(a \lor -b \lor -\mathrm{c})$

Type 3: $(-a \lor -b \lor -\mathrm{c})$

Pattern QUBOs: Transform Clauses to QUBO

Conditions

All 7 satisfying assingments of a clause need to correspond to the lowest energy of a QUBO representing the clause

The one non-satisfying assingment of a clause needs to have a higher energy than all satisfying assignments

Goal

Find 4 x 4 QUBOs that satisfy these 2 conditions.

	a	b	c	Ancilla
a	Q_1	Q_2	Q_3	Q_4
b		Q_5	Q_6	Q_7
c			Q_8	Q_9
Ancilla				Q_{10}

Why 4 x 4? It is not possible to satisfy these conditions with a 3 x 3 QUBO.

Pattern QUBOs: Energy of an assignment

A clause has 3 variables.

$$(a \vee b \vee \mathrm{c})$$

A QUBO for this clause has at least 4 variables:

	a	b	c	Ancilla
a	Q_1	Q_2	Q_3	Q_4
b		Q_5	Q_6	Q_7
c			Q_8	Q_9
Ancilla				Q_{10}

Each 3-SAT variable assignment corresponds to **multiple** QUBO bit strings.

Example:

The 3-SAT variable assignment: a = 0, b = 0, c = 0

corresponds to

QUBO bit string 1: a = 0, b = 0, c = 0, Ancilla = 0

QUBO bit string 2: a = 0, b = 0, c = 0, Ancilla = 1

Energy of an assignment:
Minimum Energy of all the corresponding QUBO bit strings.

Pattern QUBOs: Transform Clauses to QUBOs

Example

Type 0 clause: $(a \vee b \vee \mathrm{c})$

Satisfying assignments:

1. a = 0, b = 0, c = 1
2. a = 0, b = 1, c = 0
3. a = 0, b = 1, c = 1
4. a = 1, b = 0, c = 0
5. a = 1, b = 0, c = 1
6. a = 1, b = 1, c = 0
7. a = 1, b = 1, c = 1

Must have the same (minimal) energy

(Condition 1)

Not Satisfying assignment

8. a = 0, b = 0, c = 0

Must have a higher energy (Condition 2)

Algorithm Input/Output

1. Set of values to choose: {-2, 1}
2. Result:

	a	b	c	Ancilla
a	-2	1	1	1
b		-2	1	1
c			-2	1
Ancilla				-2

- ✓ All satisfying assignments have energy: -3
- ✓ The only non-satisfying assignment has energy: 0

Create new 3SAT-to-QUBO transformations

Type 0 clause	Type 1 clause	Type 2 clause	Type 3 clause
↓	↓	↓	↓
Algorithm	Algorithm	Algorithm	Algorithm
↓	↓	↓	↓
Set of QUBO representations of the type 0 clause (many!)	Set of QUBO representations of the type 1 clause (many!)	Set of QUBO representations of the type 2 clause (many!)	Set of QUBO representations of the type 3 clause (many!)

New transformation: 4-Tuple (R_0, R_1, R_2, R_3,). R_i is one (arbitrarily chosen) QUBO representation from the set of QUBO representations of the respective clause type.

Pattern QUBOS: Apply Type 0-4 QUBOs

$$\Phi = (x_1 \vee x_2 \vee \mathrm{x}_3) \wedge (x_1 \vee x_2 \vee -x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee -x_4 \vee -x_3)$$

Type 0 QUBO (R_0)	**Type 1 QUBO (R_1)**	**Type 0 QUBO (R_0)**	**Type 2 QUBO (R_2)**

Combine

3-SAT QUBO

Example (Assume only 2 types of clauses)

Type 0 clause QUBOs $(a \vee b \vee \mathrm{c})$

Qubo 0 =

	a	b	c	Ancilla
a	-2	1	1	1
b		-2	1	1
c			-2	1
Ancilla				-2

Qubo 1 =

	a	b	c	Ancilla
a		2		-2
b				-2
c			-1	1
Ancilla				1

Type 1 clause QUBOs $(a \vee b \vee -\mathrm{c})$

Qubo 2 =

	a	b	c	Ancilla
a	-1	1	0	1
b		-1	0	1
c			0	1
Ancilla				-1

Qubo 3 =

	a	b	c	Ancilla
a		2		-2
b				-2
c			1	-1
Ancilla				2

Valid transformations: (QUBO 0, QUBO2), (QUBO 0, QUBO 3), (QUBO 1, QUBO2), (QUBO 1, QUBO 3)

Example (Assume only 2 types of clauses)

$$\Phi = (x_1 \vee x_2 \vee \mathrm{x}_3) \wedge (x_1 \vee x_2 \vee -x_4) \wedge (x_1 \vee x_2 \vee x_4)$$

Choose: (QUBO 0, QUBO 2) as transformation

$(x_1 \vee x_2 \vee \mathrm{x}_3)$

	x_1	x_2	x_3	Ancilla
x_1	-2	1	1	1
x_2		-2	1	1
x_3			-2	1
Ancilla				-2

$(x_1 \vee x_2 \vee -x_4)$

	x_1	x_2	x_4	Ancilla
x_1	-1	1	0	1
x_2		-1	0	1
x_4			0	1
Ancilla				-1

$(x_1 \vee x_2 \vee x_4)$

	x_1	x_2	x_4	Ancilla
x_1	-2	1	1	1
x_2		-2	1	1
x_4			-2	1
Ancilla				-2

Example (Assume only 2 types of clauses)

	x_1	x_2	x_3	A1
x_1	-2	1	1	1
x_2		-2	1	1
x_3			-2	1
A1				-2
	x_1	x_2	x_4	A2
x_1	-1	1	0	1
x_2		-1	0	1
x_4			0	1
A2				-1
	x_1	x_2	x_4	A3
x_1	-2	1	1	1
x_2		-2	1	1
x_4			-2	1
A3				-2

	x_1	x_2	x_3	x_4	A1	A2	A3
x_1	-2 -1 -2	1 + 1 + 1	1	0 + 1	1	1	1
x_2		-2 -1 -2	1	0 + 1	1	1	1
x_3			-2		1		
x_4				0 -2		1	1
A1					-2		
A2						-1	
A3							-2

Example (Assume only 2 types of clauses)

$\Phi = (x_1 \vee x_2 \vee \mathrm{x}_3) \wedge (x_1 \vee x_2 \vee -x_4) \wedge (x_1 \vee x_2 \vee x_4)$

Choose: (QUBO 1, QUBO 2) as transformation

$(x_1 \vee x_2 \vee \mathrm{x}_3)$

	x_1	x_2	x_3	Ancilla
x_1		2		-2
x_2				-2
x_3			-1	1
Ancilla				1

$(x_1 \vee x_2 \vee -x_4)$

	x_1	x_2	x_4	Ancilla
x_1	-1	1	0	1
x_2		-1	0	1
x_4			0	1
Ancilla				-1

$(x_1 \vee x_2 \vee x_4)$

	x_1	x_2	x_4	Ancilla
x_1		2		-2
x_2				-2
x_4			-1	1
Ancilla				1

Example (Assume only 2 types of clauses)

	x_1	x_2	x_3	A1
x_1		2		-2
x_2				-2
x_3			-1	1
A1				1

	x_1	x_2	x_4	A2
x_1	-1	1	0	1
x_2		-1	0	1
x_4			0	1
A2				-1

	x_1	x_2	x_4	A3
x_1		2		-2
x_2				-2
x_4			-1	1
A3				1

	x_1	x_2	x_3	x_4	A1	A2	A3
x_1	-1	2 -1 +2		0	-2	1	-2
x_2		-1	0		-2	1	-2
x_3			-1		1		
x_4				0 -1		1	1
A1					1		
A2						-1	
A3							1

Comparison

Both QUBOs can be used to calculate solutions for the 3SAT-formula:

$$\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee x_4)$$

They differ however in the amount of non-zero couplers and in their magnitude of the linear/quadratic values.

May have an impact on the solution quality.

	x_1	x_2	x_3	x_4	A1	A2	A3
x_1	-2 -1 -2	1 + 1 + 1	1	0 + 1	1	1	1
x_2		-2 -1 -2	1	0 + 1	1	1	1
x_3			-2		1		
x_4				0 -2		1	1
A1					-2		
A2						-1	
A3							-2

	x_1	x_2	x_3	x_4	A1	A2	A3
x_1	-1	2 -1 +2		0	-2	1	-2
x_2		-1	0		-2	1	-2
x_3			-1		1		
x_4				0 -1		1	1
A1					1		
A2						-1	
A3							1

Let's get into the code ...

Let's get into the code ... after the break