QUBO from a practical Operations Research perspective

Mátyás Koniorczyk

Wigner Research Centre, ~~Hungarian Academy of Sciences~~, Budapest

GPU days 2019.

2019. 11. 22

https://wigner.mta.hu/~koniorczykmatyas/qubo

Motivated by a joint work with
KRZYSZTOF DOMINO, Research Inst. for Applied Informatics, Polish Academy of Sciences
Partially supported by NFKIH K124631 and K124351

Brief summary and motivation

Our goal is to

- get a picture of problems adiabatic quantum computers can address
- via particular examples
- actually solving some of them in an existing software framework.
  (You need: a 64 bit Intel/AMD computer with Python $\geq$ 3.3, and Internet access)

We adopt a view incorporating aspects of

- operations research and
- software engineering.

QUBO

$$\begin{array}{ll} \text{min.} & y = \mathbf{x}^T Q \mathbf{x} \\ \text{s.t.} & Q \in \{0, 1\} \end{array}$$

- In general
  $$y = \mathbf{x}^T Q' \mathbf{x} + \mathbf{b}^T \mathbf{x} + c,$$
  but $x_k^2 = x_k$, so $Q := Q' + \text{diag}(\mathbf{b})$ covers this.
- Symmetric form: $Q \to (Q + Q^T)/2$
- Upper triangular (if you like):
  $Q \to (Q + Q^T)$, then zero out the lower triangle.

What's that buzz?

- Adiabatic quantum computers can solve QUBOs [1].
- NP hard[2] and NP complete[3] problems can be formulated as QUBOs.

😎 WOW! Everybody wants to get hold of one of these!

[1] Chimera graph, minor embedding, limited size

[2] Non-deterministic polynomial-time hardness: at least as hard as the hardest in NP, that is, decision problems where "yes" instances can be verified in in polynomial time.

[3] NP and NP hard

What's that buzz?

- Many instances of hard problems can be solved with efficient classical algorithms.
- Even if not exactly, there are many good approximations and heuristics.
  (column generation, branch and bound, tabu search and other metaheuristics. . .)

☹ How should I convince experts to buy one?

- Note: on the other hand, even polynomial-time problems can be huge.

Example 1: Max cut

Problem

Given a graph $G(V, E)$ split V into two so that the no. of edges between the parts is maximal.

- NP-complete
- APX-hard: no polynomial-time approximation scheme (PTAS), arbitrarily close to the optimal solution unless P $=$ NP.
- Applications: physics: Ising model, engineering: VLSI circuit design

Example 1: Max Cut, QUBO

- $\forall j \in V$, $x_j = 0$ if in one part, $x_j = 1$ if in the other.
- $x_i + x_j - 2x_i x_j = 1$ iff $(i, j) \in E$ is in the cut.
- So we have

$$\min \sum_{(i,j) \in E} 2x_i x_j - x_i - x_j$$

Example 1: Max Cut with Ocean Toolkit

Ocean Toolkit:

`https://ocean.dwavesys.com`

This talk:

`https://wigner.mta.hu/~koniorczykmatyas/qubo`

Example 1: Max Cut with Ocean Toolkit

Lessons to learn

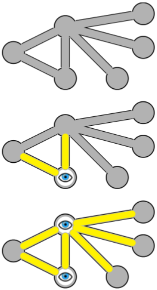
- PRO: It is easy to call a QUBO solver in Python.
- CON: This problem was just an Ising model.
- CON: We had to put together the QUBO ourselves.
  This is not what OR modelers like...

Example 2: Minimum vertex cover

Problem statement

Given a graph $G(V, E)$, c $C \subset V$ is a vertex cover if

$$(\forall e \in E)\,(\exists v \in C) \quad v \in e$$



Here the minimum vertex cover has 2 vertices.

Example 2: Minimum vertex cover

Facts

- NP-hard (decision version: NP-complete)
- Equivalent to problems with real applications.
- Integer programming (IP) formulation exists

Example 2: Minimum vertex cover: 0-1 LP formulation

Variables

$$(\forall j \in V) \quad x_j \in \{0,1\}: \ 1 \text{ iff } k \in C$$

0-1 program

$$\begin{array}{llll} \text{min.} & y & = \sum_{i \in V} x_i & \\ \text{s.t.} & x_i + x_j & \geq 1 & (\forall (i,j) \in E) \\ & x_i & \in \{0,1\} & (\forall i \in V) \end{array}$$

Suitable for CPLEX, glpk, and many more!

Remarks: About linear programming

- Non-integer: polyhedra, simplex, interior point
- Mixed integer: logic problems
- Relaxations, branch-and-bound
- Lot of effort in many solvers
  - Paid: CPLEX
  - Free: glpk, Coin-OR
  - c.f. https://en.wikipedia.org/wiki/List_of_optimization_software

Turning 0-1 programs into QUBOs

Penalties

Classical constraint	Equivalent penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + 2xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

Table quoted from F. Glover *et al.*, arXiv:1811.11538

Penalties are common in OR to deal with *soft constraints* otherwise.

QUBO for min vertex cover

$$\begin{array}{llll} \text{min.} & y & = \sum_{i \in V} x_i & \\ \text{s.t.} & x_i + x_j & \geq 1 & (\forall (i,j) \in E) \\ & x_i & \in \{0,1\} & (\forall i \in V) \end{array}$$

Classical constraint	Equivalent penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + 2xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1 x_2 + x_1 x_3 + x_2 x_3)$
$x = y$	$P(x + y - 2xy)$

$$\begin{array}{lll} \text{min.} & y & = \sum_{i \in V} x_i + P\left(\sum_{(i,j) \in E} (1 - x_i - x_j + x_i x_j) \right) \\ \text{s.t.} & x_i & \in \{0,1\} \quad (\forall i \in V) \end{array}$$

Example 2: Minimum vertex cover: Ocean Toolkit

Let's do it.

code: `02_vertex_cover.py`

Lessons to learn

- We can do 0-1 LP-s: a *tremendous* number of problems can be attacked. Their list includes:
  - Constraint programming,
  - Scheduling problems,
  - Set covering and partitioning problems,
  - Max clique search in graphs, and many other graph problems,
  - Bin packing,

  and many more, with strong *practical* applications in
  - Manufacturing,
  - Transportation and logistics,
  - Investment optimization,
  - etc.

The question is, which are the *killer applications* in which such a system is worth its price.

Lessons to learn

- Ocean Toolkit adopts a smart approach: it accepts even Networkx graphs directly.
- Check out also the constraint programming examples (e.g. `https://docs.ocean.dwavesys.com/en/latest/examples/scheduling.html`).
- All the details
  (splitting into small parts, minor embedding,
  where the dog is buried from the research perspective)
  are hidden from the user.

I want to try dWave

“Leap”: free access provided.

https://www.dwavesys.com/take-leap

Real quantum, as opposed to Sinclair’s 1984 “Quantum Leap”:

image source: https://hu.wikipedia.org/wiki/Sinclair_QL

Used and recommended literature

- A tutorial on formulating QUBOs: F. Glover *et al.*, arXiv:1811.11538 and references therein.
- Ocean documentation: `https://docs.ocean.dwavesys.com/en/latest/index.html`
- Further code examples: `https://github.com/dwave-examples`
- On model building and typical problems in OR: H. Paul Williams: Model building in mathematical programming, 5. ed. Wiley, 2013, ISBN: 978-1-118-50617-2
- Scheduling theory (partly my field; a huge zoo of hard problems): M.L. Pinedo: Scheduling. Theory, Algorithm, and Systems, Springer 2012. DOI: 10.1007/978-1-4614-2361-4