Will QC help us to solve dano3mip?

Thorsten Koch

Shanwen Pu

Yuji Shinano

Chair of
Software and Algorithms
for Discrete Optimization

ZUSE INSTITUTE BERLIN

Department:
Applied Algorithmic
Intelligence Methods

Mathematical Programming 68 (1995) 213–237

Computational experience with a difficult mixed-integer multicommodity flow problem☆

D. Bienstock*, O. Günlük

Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA

Received 27 April 1993; revised manuscript received 17 June 1994

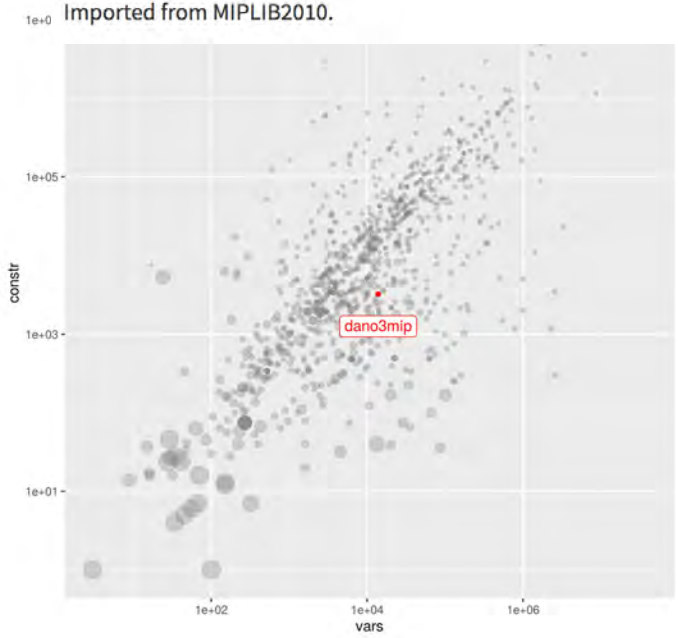
dano3mip

variable_bound | cardinality | mixed_binary

Submitter	Variables	Constraints	Density	Status	Group	Objective	MPS File
Daniel Bienstock	13873	3202	1.79317e-03	open	dano	665.571428571428*	dano3mip.mps.gz

Telecommunications applications

Imported from MIPLIB2010.



Variables	13873
Constraints	3202
Binaries	552
Integers	0
Continuous	13321
Implicit Integers	0
Fixed Variables	0
Nonzero Density	0.00179317
Nonzeroes	79655

MIPLIB 3.0

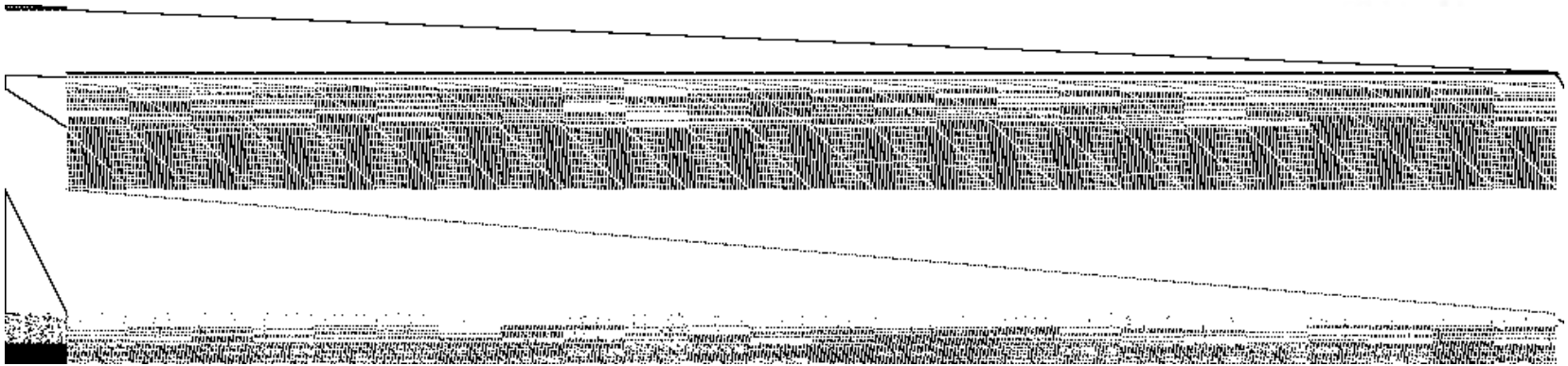
January 1996

: (To download in Netscape, click while pressing the SHIFT key)

- The MIPLIB 3.0 Problem Set as a compressed tar file
- MIPLIB Problem Set:
  - 10teams
  - air03
  - air04
  - air05
  - arki001
  - bell3a
  - bell5
  - blend2
  - cap6000
  - dano3mip
  - danoint
  - dcmulti

Best solutions so far

	ID	Objective	Exact	Int. Viol	Cons. Viol	Obj. Viol	Submitter	Date	Description
3	4	665.5714	665.5714	0	0	0	Yuji Shinano	2020-04-16	Obtained with ParaSCIP in 2014
1	3	667.5577		0	0	0	Edward Rothberg	2019-12-13	Obtained with Gurobi 9.0
4	2	676.5630		0	0	0	Robert Ashford and Alkis Vazacopoulus	2019-12-18	Found using ODH|CPlex
2	1	691.8961	691.8961	0	0	0	-	2018-10-12	Solution found during MIPLIB2017 problem selection.



$$\min z \qquad k, i, j \in \{1, \ldots, 24\}$$

$$\text{s.t.} \quad \sum_{j \neq i} x_{ij} = 2 \quad \text{for all } i \quad (1)$$

$$\sum_{j \neq i} x_{ji} = 2 \quad \text{for all } i \quad (2)$$

$$f_{kij} \leq M \cdot x_{ij} \quad \text{for all } k, i \neq j \quad (3)$$

$$\sum_{j \neq i} f_{kij} - \sum_{j \neq i} f_{kij} = s_i^k \quad \text{for all i}, k \quad (4)$$

$$\sum_{k} f_{kij} \leq z \quad \text{for all } i \neq j \quad (5)$$

$$f_{kij} \geq 0 \quad \text{for all } k, i \neq j \quad (6)$$

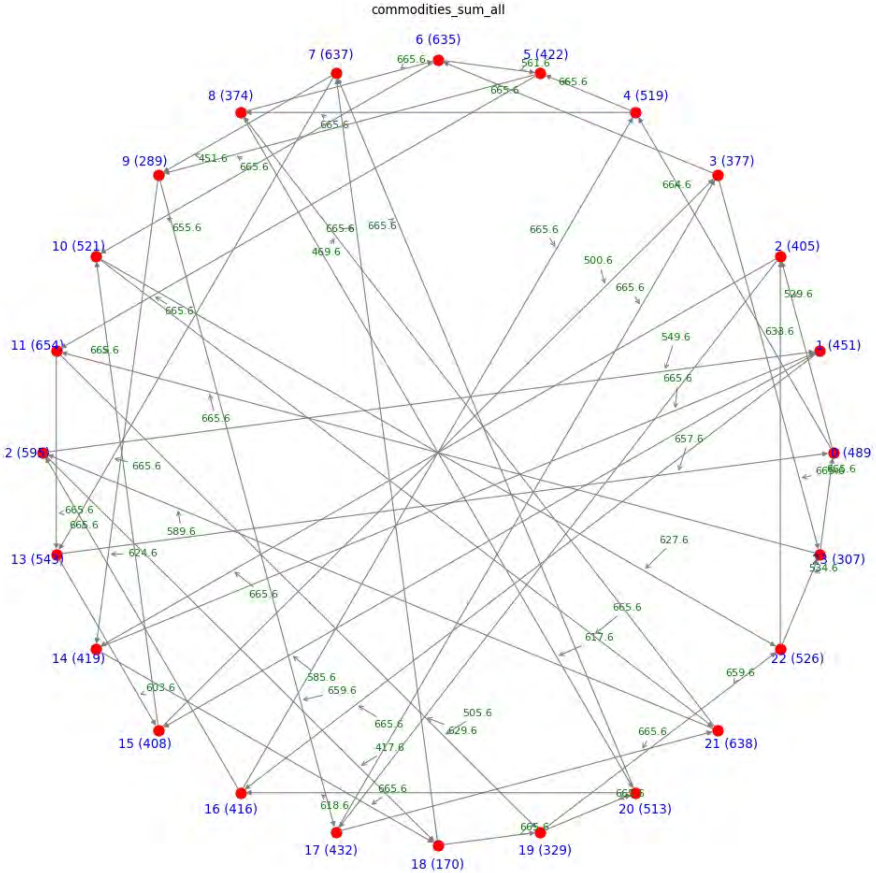
$$x_{ij} \in \{0,1\} \quad \text{for all } i \neq j \quad (7)$$

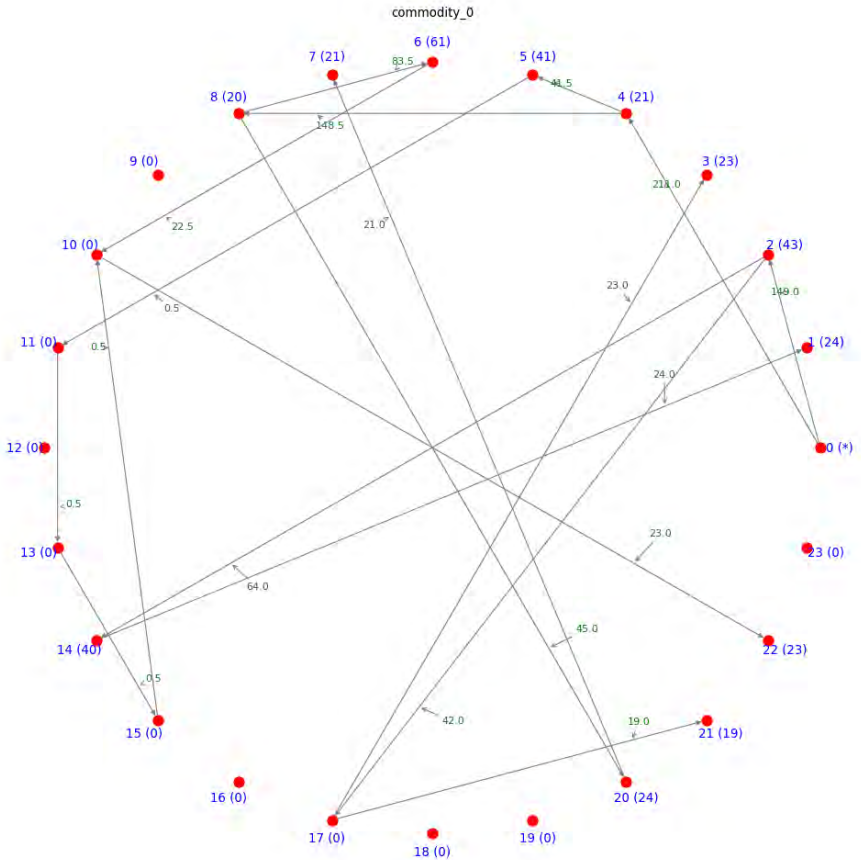
```
Root relaxation objective:  30.28571
dano3mip root relaxation : 576.2316

Best known solution:       665.5714
```

$M = 666$

```
                                30.28571   95.4%
                               350.63706   47.3%
                               425.04748   36.1%
                               436.46971   34.4%
Gurobi 9.5.1 root cuts         439.96568   33.9%

original model                 577.82468     _
with some cuts removed         577.88492     _
```

Best know solution





What are QUBOs?

QUBO : Quadratic Unconstraint Binary Optimization
UBQP : Unconstrained Binary Quadratic Program
(BIQ : Binary Integer Quadratic problem)

$$\min_{x \in \{0,1\}^n} x^T Q x$$

- x is a vector of binary variables, Q is a square $n \times n$ matrix of constants
- Since QUBOs are unconstraint, any 0/1 vector is a feasible solution
- All QUBOs can be brought to the form where Q is symmetric or upper triangular
- Solving QUBO (in general) is NP-hard
- Since x is binary, $x_i = x_i^2$ holds $\Longrightarrow$ The coefficients of the linear terms of the objective function correspond to the diagonal entries of Q

Ising Hamiltonian

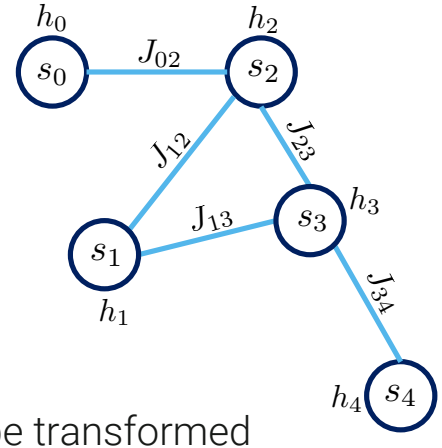
- Quantum annealing is a special-purpose device that finds the minimum energy of an Ising Hamiltonian heuristically

Linear terms (biases, local fields)

Quadratic terms (couplers)

Spins

$$H_{\text{Ising}} = \sum_{i \in V} h_i s_i + \sum_{\{i,j\} \in E} J_{ij} s_i s_j, \quad s_i \in \{-1, +1\}$$



- A Quadratic Unconstrained Binary Optimization (QUBO) problem can be transformed into an Ising Hamiltonian with a simple algebraic manipulation:

$$x^T Q x \xrightarrow[s_i \in \{-1,1\}]{x_i = \frac{s_i + 1}{2}} H_{\text{Ising}}$$

BIP

$$\min_{x \in \{0,1\}^n} c^T x$$
$$\text{s.t. } Ax \leq b$$

QUBO

$$\min_{x \in \{0,1\}^n} c^T x^2 + P(Ax + Is - b)^T (Ax + Is - b)$$

$$\min_{x \in \{0,1\}^n} x^T Q x$$

where $Q \in \mathbb{R}^{n \times n}$ and symmetric

BIPs can be reformulated as QUBOs by putting the constraints into the objective with a penalty term P. The penalty should be zero if and only if the constraint is fulfilled.

Glover, Kochenberger, Du (2019):
A Tutorial on Formulating and Using QUBO Models
arXiv:1811.11538

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

Table of a few Known constraint/penalty pairs

Independent (stable) Set

Given a graph $G = (V, E)$, find the maximum size independent set of nodes

$$\max_{S \subseteq V} |S| \text{ with } u, v \in S \Longrightarrow (u, v) \notin E$$

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

Table of a few Known constraint/penalty pairs

Binary Linear Programing formulation:

$$\max_{x \in \{0,1\}^{|V|}} \sum_{v \in V} x_v \quad \text{subject to} \quad x_u + x_v \leq 1 \quad \text{for all } (u, v) \in E$$

Unconstraint Quadratic Binary Programing formulation:

$$\min_{x \in \{0,1\}^{|V|}} -\sum_{v \in V} x_v^2 + P \cdot \underbrace{\sum_{(u,v) \in E} x_u \cdot x_v}_{=0}$$

Graph formulation $G = (V, E, w)$

$$\max_{S,T} \sum_{i \in S, j \in T} w_{ij} \quad \text{with } S \subset V, T \subset V, S \cap T = \emptyset, S \cup T = V$$

Classical Constraint	Equivalent Penalty
$x + y \le 1$	$P(xy)$
$x + y \ge 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \le y$	$P(x - xy)$
$x_1 + x_2 + x_3 \le 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

Table of a few Known constraint/penalty pairs

Binary Linear Programing formulation:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} z_{ij}$$
$$z_{ij} \le x_i + x_j$$
$$z_{ij} \le 2 - (x_i + x_j)$$
$$x_k \in \{0,1\}$$
$$z_{ij} \in \{0,1\}$$

Binary Quadratic Programing formulation:

$$\max_{x \in \{0,1\}^n} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(x_i + x_j - 2x_{ij})$$

Can be written as:

$$\min_{x \in \{0,1\}^n} x^T Q x$$

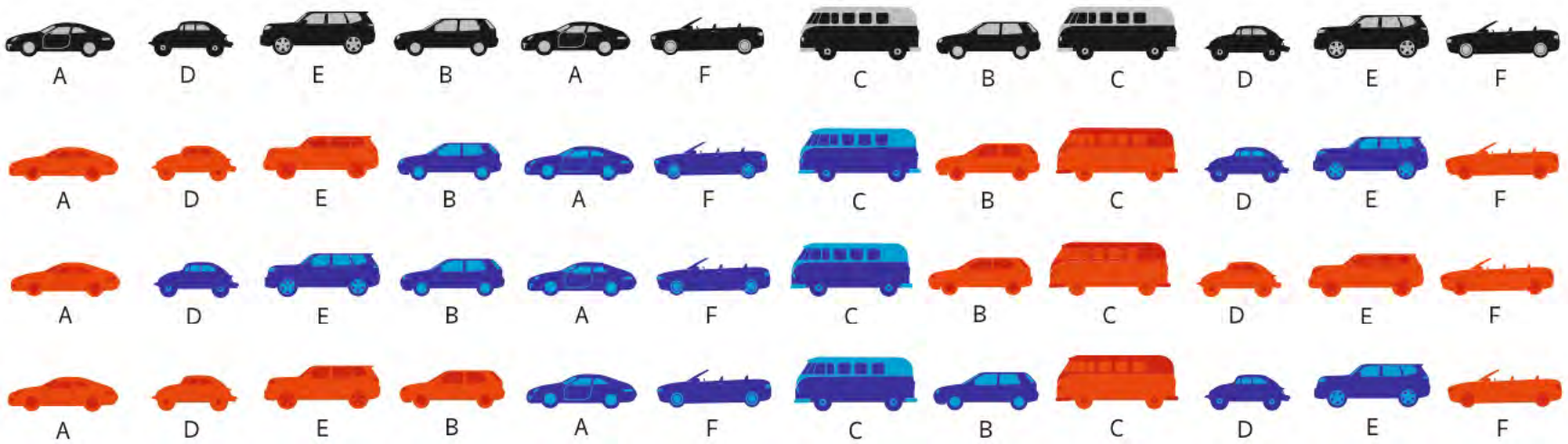
The binary paint-shop problem (BPSP)

given a word on N characters of length 2N where every character occurs exactly twice.

The objective is to color the letters of the word in two colors, such that each character receives both colors, and the number of color changes of consecutive letters is minimized.

The problem is NP-complete.

Note, if we write the solution as a 0/1 vector the number of 1s in the solution is known in advance.



Let $I := \{1, \dots, N\}$. Given a sequence $S, |S| = N$ with elements $s_i \in \{1, \dots, \frac{N}{2}\}, i \in I$ where each number appears exactly twice in s_i. We define binary variables $x_i, i \in I$.

$$\min \sum_{i=2}^{N} x_{i-1} + x_i - 2x_{i-1}x_i$$

subject to

$$x_i + x_j = 1 \text{ forall } i, j \text{ with } s_i = s_j$$

Note the quadratic term only appearing N times.

x_{i-1}	x_i	$-2x_{i-1}x_i$	=
0	0	0	0
1	0	0	1
0	1	0	1
1	1	-2	0

Now we notice that the constraints define $x_i = 1 - x_j$. This means, we can replace (substitute) all occurrences of x_i by $1 - x_j$. **This effectively removes all constraints.**
Therefore, we can directly write the problem as:

$$\min_{x \in \{0,1\}^n} x^T Q x$$

Beating a stupid opponent is easy

Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm
Sreif, Yarkoni, Skolik, Neukart, Leib (2020), arXiv:2011.03403v1

There have been several publications from VW about using a QC for solving the binary paint-shop problem, and in particular the above paper. While BPSP is NP-complete, Greedy is not a particular good heuristic for this problem.

As you can see below, a simple annealing heuristic (QUBOWL) gives very good results fast.

Sequence length	Greedy Switches	QUBOWL switches	QUBOWL time [s]	Gurobi time [s]
100	25.2	**15.6**	0.05	0.05
200	52.4	**31.6**	0.05	0.27
500	128.2	**71.8**	0.05	68.81
1,000	253.0	139.2	0.15	

We randomly generated 5 instances each of length N = 100, 200, 500, 1.000

2 x 14 core Intel Xeon Gold 6132 at 2.6-3.7 GHz used for the computation.
Gurobi 9.5.1 used for the QBP.

Time(QUBOWL) is until best solution was found.
Time(Gurobi) is until optimality was proven.

Proving optimality for length 1.000 (and bigger) is difficult with an out-of-the-box approach.
Finding an optimal or nearly optimal solution is not.

Comparing Quantum annealer with our (standard) annealing heuristic

Solving large break minimization problems in a mirrored double round-robin tournament using Quantum annealing
Kuramata, Katsuki, Nakata, arXiv:2110.07239v2 using a D-WAVE Advantage System

Table 6: **QA vs. IP in MDRRTs**

	QA		IP(Urdaneta)			QUBOWL		MR(1t)
Teams	Breaks	Time(s)	Breaks	Time(s)	OPTIMAL	Breaks	Time(s)	OPT(1h)
4	6.0	0.05	6.0	0.033974	1.0	6.0	0.05	1.0
8	19.6	0.05	19.6	0.063883	1.0	19.6	0.05	1.0
12	38.8	0.05	38.8	0.157146	1.0	38.8	0.05	1.0
16	66.0	0.05	66.0	0.681240	1.0	66.0	0.05	1.0
20	106.8	0.05	106.8	3.449914	1.0	106.8	0.15	1.0
24	161.6	0.05	156.4	52.528646	1.0	156.4	0.15	1.0
28	224.8	0.05	214.0	252.368946	0.2	**213.6**	0.15	1.0
32	280.4	0.05	267.2	288.295851	0.2	267.2	0.15	1.0
36	368.8	0.05	346.0	300.026792	0.0	**343.6**	0.75	1.0
40	453.6	0.05	422.4	300.032120	0.0	**414.0**	0.45	0.2
44	553.6	0.05	520.8	300.024345	0.0	**490.5**	0.65	0.2
48	663.6	0.05	618.8	300.024338	0.0	**565.0**	1.45	0.2

Modelling the break minimization problem for an (M)DRRT is essentially similar to the binary paint-shop problem.

Instances from the paper. *Breaks* is average number of breaks. *MR(1t)* is our Max Cut solver running on 1 thread. QUBOWL heuristic running on 2 x Intel Xeon Gold 6132 with 14 cores each at 2.6-3.7 GHz. *Time(QUBOWL)* is seconds until best solution was found.

Transforming ICONG(2) into a QUBU (so we can solve on a QC)

Todo:

1. Make it all integer
2. Make it equality ($Ax = b$)
3. Make it binary
4. Put the constraints into the objective

Make it all integer

$$\min z$$

$$\text{s.t.} \quad \sum_{j \neq i} x_{ij} = 2 \quad \text{for all } i$$

$$\sum_{j \neq i} x_{ji} = 2 \quad \text{for all } i$$

$$f_{kij} \leq M \cdot x_{ij} \quad \text{for all } k, i \neq j$$

$$\sum_{j \neq i} f_{kij} - \sum_{j \neq i} f_{kij} = \mathbf{1000} \cdot s_i^k \quad \text{for all i}, k$$

$$\sum_{k} f_{kij} \leq z \quad \text{for all } i \neq j$$

$$f_{kij} \in \{\mathbf{0}, \ldots, \mathbf{1000}\} \quad \text{for all } k, i \neq j$$

$$x_{ij} \in \{0,1\} \quad \text{for all } i \neq j$$

```
Scale by 1000 for 3 decimal digits

Root relaxation objective 3028.571
```

Transforming an ILP to a system of equations

$$\min_{x\in\mathbb{Z}} -2x_1 - x_2 \quad \text{subject to}$$

$$\begin{array}{rcrcr}
80x_1 & - & 10x_2 & \leq & -7 \\
x_1 & + & 20x_2 & \leq & 120 \\
-5x_1 & - & 20x_2 & \leq & -32 \\
7x_1 & + & 2x_2 & \leq & 48 \\
-2x_1 & + & 2x_2 & \leq & -7 \\
3x_1 & + & 4x_2 & \geq & 5 \\
3x_1 & + & 4x_2 & \leq & 5 \\
x & & & \leq & l \\
x & & & \geq & u
\end{array}$$

$\Leftrightarrow$

$$\min_{x\in\mathbb{Z}} -2x_1 - x_2 \quad \text{subject to}$$

$$\begin{array}{rcrcrcr}
80x_1 & - & 10x_2 & + & s_1 & = & -7 \\
x_1 & + & 20x_2 & + & s_2 & = & 120 \\
5x_1 & + & 20x_2 & - & s_3 & = & 2 \\
7x_1 & + & 2x_2 & + & s_4 & = & 48 \\
2x_1 & - & 2x_2 & - & s_5 & = & 7 \\
3x_1 & + & 4x_2 & & & = & 5 \\
 & & l & \leq & x & \leq & u \\
 & & & & s & \geq & 0
\end{array}$$

$$\begin{aligned}
&\min c^T x \\
&\text{s.t.} \quad Ax \leq b \\
&\qquad l \leq x \leq u \\
&\qquad x \in \mathbb{Z}^m
\end{aligned}$$

$\Leftrightarrow$

$$\begin{aligned}
&\min c^T x \\
&\text{s.t.} \quad Ax + I_m s = b \\
&\qquad l \leq x \leq u \\
&\qquad x \in \mathbb{Z}^m \\
&\qquad s \geq 0 \\
&\qquad s \in \mathbb{R}^m
\end{aligned}$$

Converting to equation form adds m variables.

$$A \in \mathbb{R}^{m\times n}, c \in \mathbb{R}^n, b, l, u \in \mathbb{R}^m$$

Making it binary

Modelling general Integer variables $z \in \{0, \dots, N\}$ using Binary variables $x_i \in \{0,1\}$:

$\sum_{i=1}^{N} x_i$	$\mathcal{O}(N)$	representation is not unique
$\sum_{i=1}^{N} i \cdot x_i \quad \wedge \quad \sum_{i=1}^{N} x_i \leq 1$	$\mathcal{O}(N)$	we need an extra constraint, which quadratic looks like $\sum_{i,j \in \{1,\dots N\}, i \neq j} x_i \cdot x_j$, and is dense
$\sum_{i=1}^{n:=\lceil\sqrt{N}\rceil} i \cdot x_i + \sum_{i=1, i \in R_1}^{n-1} i \cdot x_{n+i}$	$\mathcal{O}(\sqrt{N})$	representation is not unique
$\sum_{i=0}^{n:=\lfloor \log_2 N \rfloor} 2^i \cdot x_i + \sum_{i=1, 2^i \in R_2}^{n-1} 2^i \cdot x_{n+i}\, N$	$\mathcal{O}(\log_2 N)$	switching from $2^n - 1$ to 2^n changes all involved variables, only unique for powers of 2

with $R_1 := \{ i \in \{1, \dots, n-1\} | \sum_{i \in R} i = N - \sum_{i=1}^{n:=\lceil\sqrt{N}\rceil} i \wedge |R| \text{ minimal}\}$, $R_2 := \{ i \in \{1, \dots, n-1\} | \sum_{i \in R} i = N - \sum_{i=1}^{n:=\lfloor \log_2 N \rfloor} 2^i \wedge |R| \text{ minimal}\}$

Putting the constraints into the objective

$$\min z$$

s.t.

$$\sum_{j \neq i} x_{ij} = 2 \quad \text{for all } i$$

$$\sum_{j \neq i} x_{ji} = 2 \quad \text{for all } i$$

$$f_{kij} + s_{kij} - M \cdot x_{ij} = 0 \quad \text{for all } k, i \neq j$$

$$\sum_{j \neq i} f_{kij} - \sum_{j \neq i} f_{kij} - M \cdot s_i^k = 0 \quad \text{for all i}, k$$

$$\sum_{k} f_{kij} + s_{ij} - z = 0 \quad \text{for all } i \neq j$$

$$x_{ij} \in \{0,1\} \quad \text{for all } i \neq j$$

$$f_{kij} \in \{0, \dots, M\} \quad \text{for all } k, i \neq j$$

$$s_{kij} \in \{0, \dots, M\} \quad \text{for all } k, i \neq j$$

$$s_{ij} \in \{0, \dots, M\} \quad \text{for all } i \neq j$$

$$\min_{x \in \{0,1\}^n} c^T x^2 + P(Ax + Is - b)^T (Ax + Is - b)$$

Nodes, commodities: $k, i, j \in \{1, \dots, 24\}$

Integer to binary: $M = 666 \rightarrow \lfloor \log_2 666 \rfloor + 4 = 13$

for all $i \neq j \rightarrow 24 \times 23 = 552$

for all $k, i \neq j \rightarrow 24 \times 552 = 13248$

Binary variables total:

$$552 + 13 \times (13248 + 13248 + 552) = 352{,}176$$

$$\Longrightarrow Q \in \mathbb{Z}^{352176 \times 352176}$$

i.e., we need at least 352,176 qubits

The range of the coefficients in Q is at least up to

$$666^3 = 295{,}408{,}296$$

And Q will not be particular sparse!

Coefficient range for D-Wave Advantage

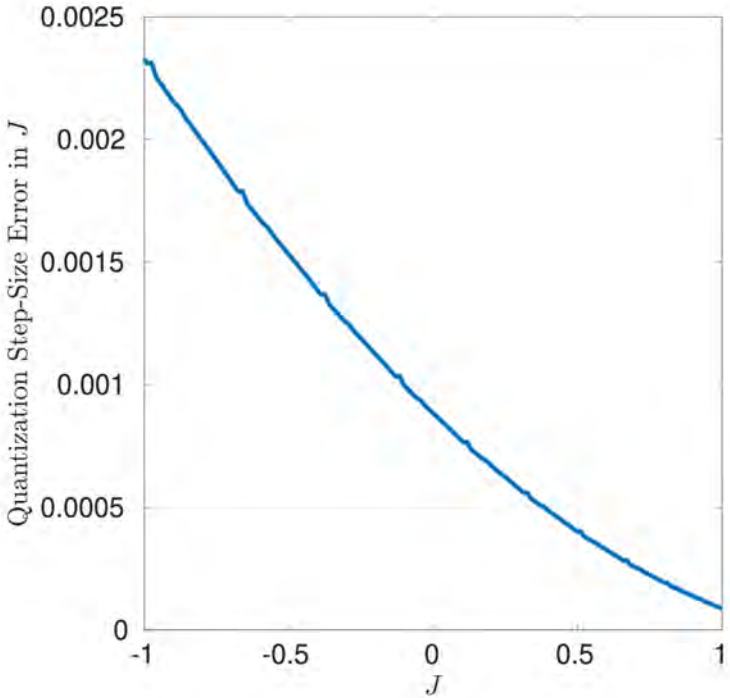


Figure 2.4: Typical quantization on the *J* DAC control.

https://docs.dwavesys.com/docs/latest/_downloads/1d0545fcfc19f50687cfb63c48d54a14/09-1264A-B_QPU_Properties_Advantage_system5_1.pdf

DAC Quantization Effects

The on-QPU digital-analog converters (DACs) that provide the user-specified h and J values have a finite quantization step size. That step size depends on the value of the h and J applied because the response to the DAC output is nonlinear. Figure 2.3 and Figure 2.4 show the effects of the DAC quantization step for the DACs controlling the h and J values, respectively, for this system.

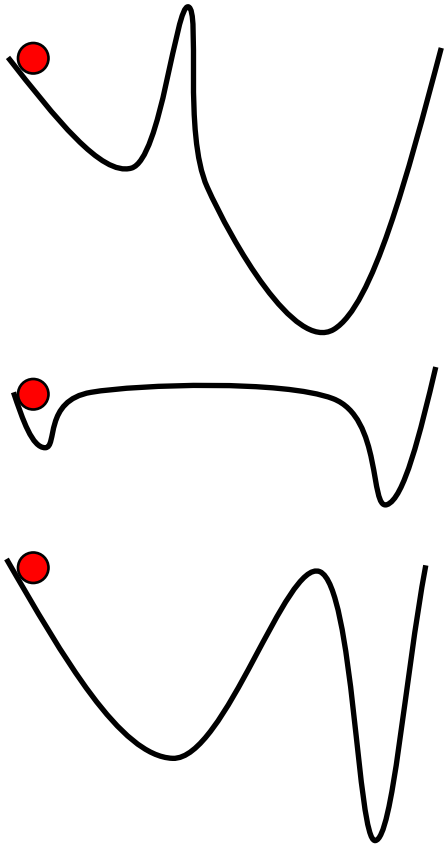
$\Rightarrow$ Dynamic range is at most $2^{11} = [0, \dots, 2047]$

ILP to QUBO

- We can transform an ILP with arbitrary integer variables into a QUBO.
- However, going from integer to binary variables will increase the problem dimension.
- Adding slacks will also increase the dimension.
- Depending on the type and number of constraints, the Q matrix might become rather dense. We are adding up all the constraints, increasing the size of the coefficients or making Q denser. Constraint with large support result in dense Q, cardinality constraints are worst.
- Since the constraints are transformed to penalties, we need a penalty factor P that grows with maximum objective value, i.e., the numeric range of the coefficients of the QUBO might be considerably larger than on the original ILP.
- Converting constraints by $(Ax - b)^T(Ax - b)$ will increase the coefficients of Q.
- The transformation loses information, as we do not know which constraints were hard anymore. This likely leads to reduced possibilities for preprocessing.

Slide by Scott Aaronson: https://www.scottaaronson.com/talks/speedup.ppt

Landscapeology



Adiabatic algorithm can find global minimum exponentially faster than simulated annealing **(though maybe other classical algorithms do better)**

Simulated annealing can find global minimum exponentially faster than adiabatic algorithm (!)

Simulated annealing and adiabatic algorithm **both** need exponential time to find global minimum

M. Jünger, E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G., T. Stollenwerk. 2021.
ACM J. Exp. Algorithmics 26, Article 1.9, doi: 10.1145/3459606

This is paper makes a very detailed and precise comparison with the following conclusion:

"However, we should stress the fact that exact optimization requires a lot of time to prove optimality, and thus it is not fair to compare their times with the heuristic times, but even with this additional burden, the exact algorithms are faster than D-Wave on a large portion of the sample.

[…]

It may well be (and we hope) that the exciting new quantum computer technology will make leaps in the future, but in our experiments, we have certainly not observed superior performance of quantum annealing in comparison to "classical" methods."

Faster exact solution of sparse MaxCut and QUBO problems

Daniel Rehfeldt, Thorsten Koch, Yuji Shinano

doi: 10.48550/arXiv.2202.02305

- Presolving
- Simple Domain Propagation
- Problem-specific cutting planes (optimized implementation)
- Primal heuristics
- Parallel branch-and-bound search via UG framework (still experimental).

Currently one of the the fastest solvers on sparse QUBO and Max-Cut benchmarks .

What was actually the main problem with dano3mip?

If we had a QC much more advanced than what is existing now
(seems we need 352,000 fully connected qubits), we could use it to compute primal solutions.

- However, what makes dano3mip so challenging, is the bad lower bound that refuses to increase.
- Currently, it is not clear how QC can help with that.

Therefore, the answer to "Will Quantum Computing help us to solve dano3mip?" is

*I am happily offering a nice bottle to the first person or AI who proves me wrong.

Look forward

"We tend to be too optimistic about the short run, too pessimistic about the long run."

— J. Preskill

- Practical MILP solving on digital computers got arguably faster at least 42% every year (combined hard+software) during the last 40 years. This is an exponential speed-up.
  Progress in Mathematical Programming Solvers from 2001 to 2020, K., Berthold, Pedersen, Vanaret, ZR-21-20
- Regarding our QUBO solver, there are still plenty algorithmic improvements possible. Additionally, we will add GPU-based heuristics and distributed memory parallelization to able to run up to 1 million cores.
- QC likely will evolve for some very specific applications, first likely around Quantum Simulation. This is the original application of QC. It has some strong inherit advantage compared to classical computers.

Recommended further Reading:

NP-complete Problems and Physical Reality, Scott Aaronson, arXiv:quant-ph/0502072

https://www.scientificamerican.com/article/will-quantum-computing-ever-live-up-to-its-hype

https://www.linkedin.com/pulse/quantum-computing-hype-bad-science-victor-galitski-1c?trk=public_post-content_share-article_title

https://www.scottaaronson.com/talks/speedup.ppt

https://physicsworld.com/a/conquering-the-challenge-of-quantum-optimization

https://m-malinowski.github.io/2022/03/11/forecasting-future-of-qc.html

Thanks a lot to Dan for the nice challenge ☺

(and the incredible work done in the paper)

My personal prediction is:

We will solve it (by traditional means) before you retire.

And I wish you will see one day dano3mip.mps
read into an out-of-the-box solver and it will just solve.

Thank You very much!

https://dilbert.com/strip/2012-04-17