Comparing the Performance of Learned and Handmade one-hot QUBO Models for Quantum Annealing

Florian Richoux[1,3] , **Jean-François Baffier**[2,3] , **Philippe Codognet**[3,4,5]
[1]AIST, Tokyo, Japan
[2]IIJ Research Lab, Tokyo, Japan
[3]JFLI, CNRS, Tokyo, Japan
[4]Sorbonne University, Paris, France
[5]University of Tokyo, Tokyo, Japan
florian@richoux.fr, jf@baffier.fr, codognet@is.s.u-tokyo.ac.jp

Abstract

Our aim is to facilitate the formulation and solving of constrained optimization problems in QUBO for quantum annealing. We defined in previous work a framework for automatically generating the QUBO penalties from various types of constraints, and we present in this paper a performance evaluation of this learning process by comparing the runtime performance of the learned QUBO programs with respect to the runtime performance of a handmade QUBO formulation. We perform this comparison for two well-known constrained optimization problems: the Travelling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP).

1 Introduction

The use of Quantum Annealing [Kadowaki and Nishimori, 1998; Farhi *et al.*, 2001] for solving combinatorial problems has raised a growing interest in the last years [Yarkoni *et al.*, 2022; Mohseni *et al.*, 2022], thanks to the development of quantum computers such as D-Wave systems [Bunyk *et al.*, 2014] and quantum-inspired systems such as Fujitsu Digital Annealing Unit [Aramon *et al.*, 2019]. These systems are based on the modeling of problems in QUBO (Quadratic Unconstrained Binary Optimization), which is equivalent to the Ising model [Lucas, 2014; Glover *et al.*, 2019].

A QUBO problem is defined by a vector of N binary decision variables $x_1, \cdots, x_N$ and a quadratic function over $x_1, \cdots, x_N$ to be minimized, of the form: $\sum_{i \leq j} q_{ij} x_i x_j$ Therefore, a QUBO problem can be reduced to an upper triangular $N \times N$ square matrix Q with coefficients q_{ij}.

While constrained optimization problems are expressed with *integer* decision variables subject to a set of *constraints*, QUBO is based on Boolean decision variables and has no constraints. Thus, in QUBO, one has first to select an encoding scheme to represent integers as Booleans and, second, to transform the constraints into *penalties* that are added to the objective function to minimize. The key idea is that a penalty will have its minimal value when the constraint is satisfied, leading thus to feasible solutions (solutions satisfying all constraints) for the optimization problem.

In the quantum annealing literature, the usual way to encode integers into Booleans is the *one-hot* encoding: an integer variable $x \in \{1, \ldots, k\}$ is represented by k Boolean variables x_i that have value 1 if the original variable x has value i and value 0 otherwise. We developed in previous work [Richoux *et al.*, 2023] an approach based on the one-hot encoding of integers for automatically generating a quadratic penalty (suitable for QUBO modeling) corresponding to a constraint between integer variables. This approach is general and based on the learning of (a limited number of) positive and negative candidates for the constraint satisfaction; it works for any type of constraint: linear constraints, non-linear constraints (such as the *all-different* or *permutation* constraint), etc.

Although this approach greatly helps in the formulation of the QUBO models, one question which is left open is to know if the QUBO models automatically generated by our approach have performance that similar to handmade QUBO models or not. We would like in this paper to answer this question by comparing the performance of the learned QUBO models with respect to the runtime performance of handmade QUBO formulations for two well-known constrained optimization problems: the Travelling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP).

2 Methodology

Converting a classic model to a QUBO model requires finding a Q_c matrix representation of each constraint in the original model, so that the global Q matrix representing the QUBO model of the problem would simply be the sum of each Q_c matrices with the matrix expressing the objective function. In [Richoux *et al.*, 2023], we convert each constraint c by learning Q_c matrix from data as a pattern composition over a one-hot encoding. The training data is derived from a specific instance of c, where c defines a constraint over a fixed number of variables with values from domains of fixed size.

Let c be a constraint and ι one of its possible instances. A **candidate** for ι refers to an assignment of all variables within ι. Let $Cand(\iota)$ be the set of all candidates of the constraint instance ι. A candidate is deemed **positive** if it satisfies c and **negative** otherwise. Let $S = \{(x, y) \mid x \in Cand(\iota), y \in \{0, 1\}\}$ be the set of all possible pairs (x, y),

such that:

$$y = \begin{cases} 1 & \text{if } x \text{ is a positive candidate} \\ 0 & \text{if } x \text{ is a negative candidate.} \end{cases}$$

The goal of our method is, given some positive and negative candidates in $Cand(\iota)$ provided by users, to find a pattern composition representing a Q_c matrix corresponding to c, such that the following property holds:

$$\forall (x, y) \in S, x^T Q_c x \text{ is minimal iff } y = 1 \qquad (1)$$

Since we deal with discrete constraints, Q_c can comprise only integers. For a constraint with a scope of n variables x_i in a k-ary domain, Q_c is an upper triangular matrix of size $nk \times nk$. Our approach involves learning a proper combination of submatrix patterns. Two types of submatrices are considered: $k \times k$ square submatrices, expressing properties between two decision variables x_i and x_j (with $i \neq j$), and $k \times k$ triangle submatrices, expressing properties of x_i^2 for all $i \in \{1, n\}$.

We have defined 15 square and 3 triangle submatrix patterns over one-hot encoding. To compose a Q_c matrix, submatrix patterns are combined by summing their elements. Some square submatrix patterns are mutually exclusive; for example, enforcing both $x_i = x_j$ and $x_i \neq x_j$ simultaneously is nonsensical. Submatrices and patterns exclusively pertain to binary variables. Our method learns from data a pattern composition representing Q_c within one second.

3 Performance Comparison

We compared the performance of learned and handmade QUBO penalties on two well-known Constrained Optimization Problems: the Quadratic Assignment Problem (QAP) and the Traveling Salesman problem (TSP). The classical formulation of these problems as permutation problems on integers $\{1, \ldots, k\}$ consists in an objective function to minimize ("length of "tour" for TSP and "flow of items between facilities" for QAP) and a permutation constraint. We will therefore compare the runtime performance of a QUBO model with a learned penalty for the permutation constraint with respect to a handmade QUBO model with a *two-way one-hot* permutation constraint [Glover *et al.*, 2019; Matsubara *et al.*, 2020]. We benchmark these two QUBO formulations on instances from QAPLIB[1] and TSPLIB[2].

Quantum annealing computers such as D-Wave are limited in terms of qubits and connections between qubits and cannot solve on the QPU the instances from QAPLIB and TSPLIB. Therefore, we used a *quantum-inspired / digital* annealing system: Fixstars Amplify Annealing Engine, a digital annealer running on a cluster of NVIDIA V100 GPUs (Graphics Processing Units), with a basic capacity of 65,536 Boolean variables connected by a complete graph.

Table 1 shows the results for QAP. We list, for each instance, the instance name, the value of the optimal solution, the best value obtained by the model, the average Time-To-Solution (TTS) over 10 runs if the solver always reaches the optimal solution within the timeout (1 minute) or the Average Relative Percentage Deviation (ARPD) w.r.t. the optimal value if this is not the case, for both the model with handmade and with learned permutation constraint. The performance for both models are very similar, except maybe in a very few cases (esc32a, and esc32b to a lesser extent).

Table 1: Comparison on QAP

Instance		Handmade model (one-hot)		Learned model (one-hot)	
name	opt.	best	TTS/ARPD	best	TTS/ARPD
rou12	235528	235528	**0.214**	235528	0.324
rou20	725522	726988	2.02%	726100	**0.79%**
had12	1652	1652	**0.321**	1652	0.363
had20	6922	6922	26.26	6922	**18.63**
nug12	578	578	**0.098**	578	0.106
nug20	2570	2570	28.38	2570	**26.47**
scr12	31410	31410	**0.077**	31410	0.121
chr12a	9552	9552	0.057	9552	**0.056**
chr12b	9742	9742	0.179	9742	**0.116**
chr20a	2192	2192	**0.501**	2192	1.947
chr20b	2298	2298	**0.637**	2298	1.534
bur26a	5426670	5638085	3.90%	5630010	**3.75%**
bur26b	3817852	4010070	5.03%	3996400	**4.68%**
esc16a	68	68	**0.063**	68	**0.063**
esc16b	292	292	**0.062**	292	**0.062**
esc32a	130	130	**1.798**	130	26.41
esc32b	168	168	**0.183**	168	0.801
tai17a	491812	492182	**0.08%**	492819	0.20%
tai20a	703482	710359	**0.98%**	712409	1.27%
tai30a	1818146	1870453	**2.88%**	1878280	3.31%
tai40a	3139370	3262846	3.92%	3251990	**3.59%**
tho30	149936	151310	**0.92%**	151383	0.96%
tho40	240516	246214	**2.37%**	246967	2.68%
wil50	48816	49541	1.49%	49284	**0.96%**

Table 2: Comparison on TSP

Instance		Handmade model (one-hot)		Learned model (one-hot)	
name	opt.	best	TTS/ARPD	best	TTS/ARPD
ulysses16	6859	6859	**0.312**	6859	0.468
gr17	2085	2085	0.262	2085	**0.228**
gr48	5046	5055	**0.08%**	5055	**0.08%**
fri26	937	937	**2.594**	937	5.094
att48	10628	10691	**0.59%**	10708	0.75%
hk48	11461	11480	1.65%	11521	**0.52%**
eil51	426	431	**1.17%**	433	1.64%
berlin52	7542	7739	**2.61%**	7833	3.86%

Table 2 shows the same results for the TSP. Performance of both models are similar, with a slight advantage sometimes to the handmade model and sometimes to the learned model.

Looking directly at the QUBO matrices, the models with handmade permutation constraint and the models with the learned permutation constraint are quite different, thus it was not clear *a priori* that the runtimes would be similar.

4 Conclusion

We compared the performance of learned QUBO models and handmade QUBO models, with one-hot encoding for integers, on two classical constrained optimization problems: TSP and QAP. Experiments on a quantum-inspired annealer show that both models have very similar performances. This shows that the use of QUBO penalties derived automatically by learning are a viable approach in order to simplify the design of QUBO models for quantum annealing.

[1] coral.ise.lehigh.edu/data-sets/qaplib/

[2] comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

References

[Aramon *et al.*, 2019] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut G. Katzgraber. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics*, 7:48, 2019.

[Bunyk *et al.*, 2014] Paul I. Bunyk, Emile M. Hoskinson, Mark W. Johnson, Elena Tolkacheva, Fabio Altomare, Andrew J. Berkley, Richard Harris, Jeremy P. Hilton, Trevor Lanting, Anthony J. Przybysz, and Jed Whittaker. Architectural considerations in the design of a superconducting quantum annealing processor. *IEEE Transactions on Applied Superconductivity*, 24(4):1–10, 2014.

[Farhi *et al.*, 2001] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[Glover *et al.*, 2019] Fred W. Glover, Gary A. Kochenberger, and Yu Du. Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *4OR*, 17(4):335–371, 2019.

[Kadowaki and Nishimori, 1998] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Phys. Rev. E*, 58:5355–5363, 1998.

[Lucas, 2014] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[Matsubara *et al.*, 2020] Satoshi Matsubara, Motomu Takatsu, Toshiyuki Miyazawa, Takayuki Shibasaki, Yasuhiro Watanabe, Kazuya Takemoto, and Hirotaka Tamura. Digital annealer for high-speed solving of combinatorial optimization problems and its applications. In *25th Asia and South Pacific Design Automation Conf.*, pages 667–672, 2020.

[Mohseni *et al.*, 2022] Naeimeh Mohseni, Peter L. McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Rev. Phys.*, 4(6):363–379, 2022.

[Richoux *et al.*, 2023] Florian Richoux, Jean-François Baffier, and Philippe Codognet. Learning QUBO models for quantum annealing: A constraint-based approach. In *Proceedings of the 2023 International Conference on Computational Science (ICCS)*, pages 153–167. Springer LNCS, 2023.

[Yarkoni *et al.*, 2022] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, 85(10):104001, 2022.