Exact and Sequential Penalty Weights in Quadratic Unconstrained Binary Optimisation with a Digital Annealer

Marcos Diez García
University of Exeter
Exeter, United Kingdom
md518@exeter.ac.uk

Mayowa Ayodele
Fujitsu Research of Europe Ltd.
Slough, United Kingdom
mayowa.ayodele@fujitsu.com

Alberto Moraglio
University of Exeter
Exeter, United Kingdom
a.moraglio@exeter.ac.uk

ABSTRACT

Quadratic unconstrained binary optimisation (QUBO) problems is a general class of combinatorial optimisation problems, which regained popularity after recent advances in quantum computing. Quantum-inspired technologies like Fujitsu's Digital Annealer (DA), based on simulated annealing, can solve QUBO problems much faster than traditional computers. Penalty methods can convert constrained optimisation problems into QUBO problems. However, existing *exact* methods of determining penalty weights are limited to specific QUBO problems and require manual analysis by experts in QUBO. Empirical methods are general but become computationally prohibitive for large problem sizes and do not guarantee that penalty weights preserve the feasibility of global optima. We present a simple, efficient, general method applicable to *any* QUBO to determine exact penalty weights. Such weights are simple upper-bounds of the smallest penalty weight *guaranteeing* that unconstrained global optima are the same as the feasible global optima. These bounds can be iteratively improved by sequential penalty methods which we also present. Experimental results with the DA on minimum cut, travelling salesman and multi-dimensional knapsack problems show the viability of the novel methodology hybridising exact and sequential methods. This work contributes towards general, automatic and scalable penalty methods in QUBO.

CCS CONCEPTS

• **Mathematics of computing → Optimization with randomized search heuristics**; **Combinatorial optimization**.

KEYWORDS

Quadratic unconstrained binary optimisation, constraint handling, penalty weight, simulated annealing, Digital Annealer

ACM Reference Format:
Marcos Diez García, Mayowa Ayodele, and Alberto Moraglio. 2022. Exact and Sequential Penalty Weights in Quadratic Unconstrained Binary Optimisation with a Digital Annealer. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528925

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528925

1 INTRODUCTION

Quadratic unconstrained binary optimisation (QUBO) is a widely used mathematical framework to formulate and solve unconstrained versions of many constrained combinatorial optimisation (CCO) problems [15]. Besides its numerous real-world applications [13], QUBO became an essential part of specialised computers, such as D-Wave's quantum annealer [17] and Fujitsu's Digital Annealer (DA) [16], enabling more efficient and large-scale problem solving.

CCO problems can be reformulated as QUBO problems via penalty methods [5, 13, 15]. These cast the problem's constraints as a non-negative term that is added to the objective function to increase the cost of any infeasible solution (i.e. penalise them). However, penalty methods pose a major challenge [5, 22]: it is not always clear how penalty terms should be weighted. Valid penalty weights should not be so small as to render infeasible the globally optimal solutions yet not so large as to harm search performance due to large jumps in fitness between feasible and infeasible solutions. The following are some of the most typical penalty methods.

Exact penalty methods [8, 14] model a constrained optimisation problem as an unconstrained problem, where an appropriate penalty weight *guarantees* that the unconstrained global minima are *exactly* the solutions of the constrained problem. Finding a theoretical upper-bound of such weight is possible by analysis of the problem class, the objective or penalty functions [15]. However, this is a manual and tedious approach that relies on the mathematical structure of each specific problem to obtain tight penalty weight upper-bounds; tightness being a critical performance factor [14]. So limiting the bounds to one problem class or instance. Also, such case-by-case analyses are often accessible only to experts in QUBO, and even so it becomes incredibly challenging for problems with complicated representations and constraints [9].

Unlike exact penalty methods, sequential penalty methods model a constrained optimisation problem as a *sequence* of unconstrained minimisation problems [10, 21]. That is, initially solving the problem with a very small penalty weight; then, gradually increasing the weight and solving the problem with the new weights until a *feasible* (near) optimum is obtained. In practice sequential penalty methods have been computationally limited to small problem sizes (often less than a thousand variables), and their success is sensitive to the starting weight as well as the increase rate. Alternatively, fast greedy heuristics [12] and metaheuristics like evolutionary algorithms [5, 13] were developed to tackle large-scale QUBO forms of CCO problems. Yet penalty weights are often manually chosen ad hoc per problem instance or by trial and error without provably guaranteeing their validity [9, 20].

Overall, correctly setting penalty weights raises two challenges. First, developing a general, automatic, way to find valid penalty

weights which preserve feasible global optima and have relatively small magnitude. Second, understanding more deeply how different penalty weights and types of penalty methods can affect search performance in terms of solution quality, feasibility, or runtime.

This paper addresses the above challenges by developing a novel framework for a general, automatic and efficient penalty method that can find upper-bounds of the smallest valid penalty weight given a CCO problem. So it frees us from manually guessing penalty weights, scales well as problem size grows, and is not limited to only one class of CCO problems nor instance. Our framework is based on exact penalty methods whereby such penalty weight upper-bounds can be efficiently derived using general, well-known, theoretical bounds on QUBO formulas. Since these bounds are general, the penalty weight upper-bounds may be loose. To improve on them, we also design new sequential penalty methods *informed* by such upper-bounds. Using Fujitsu's third-generation DA [18] as solver, we benchmark all these penalty methods on well-known instances of minimum cut, travelling salesman and multi-dimensional knapsack problems [2, 7, 19]. We find that hybridising the exact and sequential methods generally leads to solutions closer to optimal.

2 PRELIMINARIES

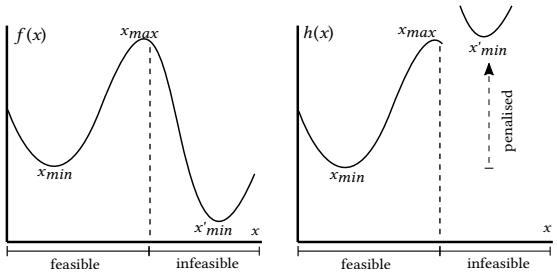
QUBO problems consist in minimising an unconstrained quadratic pseudo-Boolean function of $n \in \mathbb{N}$ binary variables $x_1, \ldots, x_n$, which can always be expressed in the following polynomial form [3]: $q(x) = c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{1 \le i < j \le n} c_{ij} x_i x_j$ with fixed scalar coefficients c_0, c_i and $c_{i,j}$. We call $q(x)$ the cost, energy or fitness of a candidate solution x. All penalty methods presented later consider unconstrained quadratic pseudo-Boolean functions $h(x) = f(x) + w \cdot g(x)$; w is a non-negative scalar, f is an objective function and g a penalty function for a corresponding CCO problem. For simplicity, we do not address converting constraints into penalty functions, as we focus on adjusting penalty weights. We assume: (a) there always is at least one feasible global optimum; (b) f and g can always be expressed in the polynomial form of $q(x)$; (c) $g(x) = 0$ if x is feasible and $g(x) \ge 1$ whenever x is infeasible; and, (d) all constraints of a given CCO problem can be aggregated into g so being equally penalised by the same weight w.

DA is a dedicated processor, introduced by Fujitsu, that is inspired in adiabatic quantum computation and uses massive parallelism to solve QUBO problems more efficiently [16]. It performs a meta-heuristic search based on simulated annealing, but DA incorporates extra mechanisms to escape from local optima and evaluate in parallel the solution quality of all bit-flip neighbours adjacent to a given solution [1]. Third-generation DA can handle QUBO sizes up to $n = 100{,}000$ [16, 18]. *The penalty methods we present use the DA but do not require it, so other search algorithms may be used.*

3 EXACT PENALTY METHODS

In QUBO formulations of a given CCO problem, we require that the unconstrained global optimum of h must be the exactly same as the feasible global optimum of the original constrained problem.

When penalty weights are "too small", global minima of h may become infeasible as infeasible solutions were not penalised enough. Sufficiently large weights avoid that, but no general notion of "large enough" has been defined. Here we quantify this in a simple, rigorous and general way. Figure 1 sketches how infeasible solutions are penalised to have higher cost than all feasible solutions (Figure 1b) including the global maximum of f, so the global minimum of h is feasible and coincides with the feasible minimum in the constrained problem (Figure 1a).



(a) Constrained problem **(b) Unconstrained problem**

Figure 1: Validity of penalty weights.

We define a penalty weight $w \ge 0$ as *valid* if and only if $h(x_{\min}) = f(x_{\min}) < h(x') = f(x') + w \cdot g(x')$ holds for all $x_{\min}, x' \in \{0,1\}^n$ where $x_{\min} = \operatorname{argmin}_{x \in \{0,1\}^n} h(x)$, $g(x_{\min}) = 0$ and $g(x') \ge 1$. Let $f_{\max}$ and $f_{\min}$ be respectively the (unconstrained) global maximum and minimum objective values. Note feasible solutions cannot have worse objective value than $f_{\max}$, and all infeasible solutions have $g(x') \ge 1$ by assumption. Then, choosing any penalty weight $w > \Delta f = f_{\max} - f_{\min}$, as depicted in Figure 1b, would satisfy our validity condition. Any upper-bound of such w, derived by upper-bounding $f_{\max}$ and lower-bounding $f_{\min}$, is valid as well.

Known techniques to derive such valid penalty weights are: (a) upper-bounding $f_{\max}$ (respectively, lower-bounding $f_{\min}$) by summing the constant coefficient and positive (respectively, negative) coefficients in all linear as well as quadratic QUBO terms [3, 4]; (b) using the upper (respectively, lower) bounds given by the constant term in negaforms (respectively, posiforms) of f in QUBO form [3, 4]; and (c) via Verma and Lewis's [23] technique with quadratic computational complexity in the number of non-zero QUBO terms.

4 SEQUENTIAL PENALTY METHODS

This section proposes scaled-sequential (Algorithm 2) and binary search penalty methods (Algorithm 3) as new variants of a traditional form of sequential penalty method in Algorithm 1. These variants differ from traditional sequential methods [10, 21] because they *exploit knowledge from upper-bounds of valid penalty weights*, which one may compute via the exact penalty methods (Section 3). We hypothesise this knowledge guides our new sequential methods by narrowing the sequence of penalty weights that need to be generated until a valid one is found, and thus obtain better penalty weights in fewer trials. Any solver that can find optimal or sub-optimal solutions (e.g. DA [1, 16]) can be used together with the three sequential methods to find feasible minima and small valid penalty weights. All three methods generate weights on a power scale. That is because previous research [13] suggests ballpark estimates of a target penalty weight in practice already lead good to search performance, and the magnitude order of smallest valid penalty weights can be reached in exponentially fewer trials than a linear scale. Unlike other sequential methods [5, 10], none of the sequential methods here do not require an initial feasible solution.

Algorithm 1 (Standard) Sequential Penalty Algorithm

$w \leftarrow 1$ ▷ initial penalty weight
for each iteration **do**
  minimise $h(x) = f(x) + w \cdot g(x)$ with one DA run
  if solution found is feasible **then**
    record solution found and w
  $w \leftarrow w * 10$
return minimum feasible solution found and corresponding w

Algorithm 2 Scaled-sequential Penalty Algorithm

Input: $w_U \leftarrow$ upper-bound of a valid penalty weight
Input: t, maximum number of iterations ($t > 1$)
$w \leftarrow 1$ ▷ initial penalty weight
scale_factor $\leftarrow {w_U}^{1/(t-1)}$
for each iteration **do**
  minimise $h(x) = f(x) + w \cdot g(x)$ with one DA run
  if solution found is feasible **then**
    record solution and w found
  $w \leftarrow$ round($w \cdot$ scale_factor)
return minimum feasible solution found and corresponding w

Algorithm 3 Binary Search Penalty Algorithm

Input: $w_U \leftarrow$ upper-bound of a valid penalty weight
interval endpoint $a \leftarrow 1$; interval endpoint $b \leftarrow w_U$
for each iteration **do**
  $w \leftarrow \text{round}\left(\sqrt{a \cdot b}\right)$ ▷ power scale midpoint
  minimise $h(x) = f(x) + w \cdot g(x)$ with one DA run
  if solution found is feasible **then**
    record solution found and w
    $b \leftarrow w$ ▷ update interval endpoint
  else
    $a \leftarrow w$ ▷ update interval endpoint
return minimum feasible solution found and corresponding w

5 EXPERIMENTAL SETTINGS

In our experiments we use the DA and known QUBO formulations, derived via standard reformulation techniques [13], for the following CCO problems: (a) the minimum cut (Mincut) problem, with a binary representation and single logical constraint [13, 16]; (b) the travelling salesman problem (TSP), with a permutation representation and multiple logical constraints [15]; and (c), the multi-dimensional 0-1 knapsack problem (MKP), with a binary representation as well as equality and logical constraints [6]. Here the penalty functions meet the assumption $g(x) \geq 1$ for infeasible solutions x, as required by our framework. For reproducibility, the used QUBO formulas are available as separate Python's Numpy NPZ files in an online repository [11]. A total of 20 problem instances taken from well-known benchmark suites [2, 7, 19] will be tested, ranging in QUBO size from $n = 50$ (MKP *weing1*) up to $n = 4960$ (Mincut *add32*) decision variables.

The DA has two stopping criteria: *target energy* (i.e. cost to be minimised) and *time limit*. We set *target energy* to a known optimal for each problem and *time limit* to 20 seconds. DA stops its run if it reaches the known optimal before 20 seconds, otherwise it stops once 20 seconds of solve time elapse. DA is executed independently 20 times for each set of experiments. The sequential methods (Section 4) use a maximum of 10 iterations. The experiments stop before 10 iterations if: (1) feasibility is reached using the standard or scaled sequential methods; or, (2) the interval endpoints in the binary search method reduce to a single penalty weight. For w_U, Algorithms 2–3 will use sum of coefficients as exact penalty method.

To measure solution quality with respect to a known optimal solution, we use the *average relative percentage deviation* (ARPD). This expresses as a percentage the absolute difference between the known optimal and the solution quality found by the DA averaged over 20 runs. The ARPD is calculated based on feasible solutions.

6 RESULTS

We present results of preliminary experiments. For all of them, the feasibility rate across the 20 DA runs was 100%. We note all penalty weights found by all exact methods satisfied our validity condition (Section 3), so preserving feasibility of global optima, but they were not always the smallest compared with best known weights per problem instance. This is expected since all exact methods, by design, produce general upper-bounds of such smallest penalty weights. Among them, Verma and Lewis's technique [23] ("Verma-Lewis") obtained smaller weights than the posiform-negaform ("Posi-Nega") and sum of coefficients ("Sum") methods. Though Verma-Lewis was approximately two or three orders of magnitude slower than Posi-Nega and Sum across the problem instances. This is expected since Posi-Nega and Sum are simpler methods. This also agrees with penalty weights by Verma-Lewis leading to better solution quality on Mincut instances on average, and marginally better on MKP instances, compared with Posi-Nega and Sum (Table 1). There is negligible difference between the exact methods for TSP. By contrast, all sequential methods (on average) attained solutions closer to optimal within 10 iterations when compared with the exact methods on Mincut and MKP instances (Table 2). Particularly, our proposed scaled-sequential and binary search methods always attained optimal solutions within 10 iterations across all Mincut instances, unlike the standard sequential method. For MKP and TSP instances there were not significant differences between sequential methods. But all of them led to solutions within 13.20% of optimal in 10 iterations or less; the scaled-sequential and the binary search methods were marginally better than the (uninformed) standard sequential penalty method. We expect the performance superiority of the informed sequential methods, especially binary search, to stand out over the standard sequential as well as exact methods on a more diverse set of problems and a suitable experimental setting.

7 CONCLUSIONS

We presented a general, scalable, framework to adjust penalty weights for *any* QUBO formulation of CCO problems while *guaranteeing* feasibility of global optima. This framework is based on exact and sequential penalty methods as well as known theoretical bounds on QUBO formulas. We analysed experimentally how penalty weights by these methods affect third-generation DA's performance on QUBO instances of Mincut, MKP and TSP. We show that hybridising a simple exact method with scaled or binary search

sequential methods leads to solutions closer to optimal for many instances. We plan to benchmark our methods on more diverse problem sets, use tighter theoretical bounds on the objective function's QUBO form or exploit information from the constraint functions.

Table 1: Average relative percentage deviation achieved by the DA with the exact penalty methods

Problem Category	Instance Name	Optimal	ARPD		
			Posi-Nega	Sum	Verma-Lewis
Mincut	add20	596	0.00	0.00	0.00
	data	189	0.00	5.29	0.00
	3elt	90	26.67	26.67	0.00
	uk	20	190.00	190.00	0.00
	add32	11	809.09	809.09	100.00
		Average	205.15	206.21	**20.00**
MKP	weing1	141,278	0.00	0.00	0.00
	weing3	95,677	0.00	0.00	0.00
	weing5	98,796	0.23	0.26	0.00
	weing7	1,095,445	0.24	0.27	0.25
	weing8	624,319	1.00	1.00	1.02
	weish01	4,554	3.14	3.14	0.64
	weish06	5,557	1.69	1.69	2.50
	weish12	6,339	6.42	3.37	5.36
	weish18	9,580	6.20	6.20	5.91
	weish30	11,191	5.58	4.63	3.08
		Average	2.45	2.05	**1.88**
TSP	fri26	937	0.00	0.00	0.00
	bays29	2020	0.00	0.00	0.00
	dantzig42	699	0.72	0.72	0.72
	brazil58	25395	3.16	3.16	3.16
	st70	675	3.66	3.66	3.67
		Average	**1.51**	**1.51**	**1.51**

Table 2: DA results (20 runs per iteration): Best ARPD derived using the sequential methods of updating penalty weights

Problem Category	Instance Name	Optimal	ARPD using the minimum penalty weight			Best ARPD in 10 iterations		
			Standard	Scaled	Binary Search	Standard	Scaled	Binary Search
Mincut	add20	596	0.50	0.50	0.50	0.00	0.00	0.00
	data	189	0.00	0.00	0.00	0.00	0.00	0.00
	3elt	90	0.00	0.00	0.00	0.00	0.00	0.00
	uk	20	0.00	0.00	0.00	0.00	0.00	0.00
	add32	11	54.55	54.55	54.55	54.55	0.00	0.00
		Average	11.01	11.01	11.01	10.91	**0.00**	**0.00**
MKP	Weing1	141,278	1.61	9.79	14.02	0.00	0.00	0.00
	Weing3	95,677	17.24	10.14	29.09	0.00	0.00	0.00
	Weing5	98,796	5.96	31.12	31.12	0.00	0.00	0.00
	Weing7	1,095,445	5.21	3.80	3.35	0.09	0.02	0.06
	Weing8	624,319	0.09	23.00	22.71	0.06	0.39	0.70
	Weish01	4,554	3.93	3.93	3.93	1.62	0.00	0.00
	Weish06	5,557	14.54	14.54	14.54	0.00	0.27	0.50
	Weish12	6,339	39.53	39.53	39.53	0.48	0.02	0.00
	Weish18	9,580	8.04	8.04	8.04	0.72	0.37	0.51
	Weish30	11,191	32.97	32.97	33.10	0.47	0.29	0.29
		Average	12.91	17.69	19.94	0.34	**0.14**	0.21
TSP	fri26	937	8.00	0.00	13.77	1.54	0.00	0.00
	bays29	2020	0.40	0.79	2.77	0.40	0.40	0.00
	dantzig42	699	1.29	3.15	8.73	1.29	3.15	0.00
	brazil58	25395	13.20	11.83	19.06	13.20	11.83	8.03
	st70	675	6.21	2.87	9.48	6.21	2.87	9.25
		Average	5.82	3.73	10.76	4.53	3.65	**3.46**

REFERENCES

[1] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut G. Katzgraber. 2019. Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer. *Frontiers in Physics* 7, 48 (2019), 1–14. Frontiers. https://doi.org/10.3389/fphy.2019.00048.

[2] John E. Beasley. 1990. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society* 41, 11 (1990), 1069–1072. Springer. https://doi.org/10.1057/jors.1990.166.

[3] Endre Boros and Peter L. Hammer. 2002. Pseudo-Boolean optimization. *Discrete Applied Mathematics* 123, 1–3 (2002), 155–225. Elsevier. https://doi.org/10.1016/S0166-218X(01)00341-9.

[4] Endre Boros, Peter L. Hammer, and Gabriel Tavares. 2006. *Preprocessing of Unconstrained Quadratic Binary Optimization.* Technical Report RUTCOR Research Report RRR 10-2006. Rutgers University, New Jersey, USA.

[5] Carlos A. Coello Coello. 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 191, 11-12 (2002), 1245–1287. Elsevier. https://doi.org/10.1016/S0045-7825(01)00323-1.

[6] Mark W Coffey. 2017. Adiabatic quantum computing solution of the knapsack problem. *arXiv preprint arXiv:1701.05584* (2017).

[7] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software* 38, 1 (2011), 1–25. ACM. https://doi.org/10.1145/2049662.2049663.

[8] G. Di Pillo and L. Grippo. 1989. Exact Penalty Functions in Constrained Optimization. *Journal on Control and Optimization* 27, 6 (1989), 1333–1360. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/0327068.

[9] Sebastian Feld et al. 2019. A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer. *Frontiers in ICT* 6, 13 (2019), 1–13. Frontiers. https://doi.org/10.3389/fict.2019.00013.

[10] Anthony V. Fiacco and Garth P. McCormick. 1990. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Classics in Applied Mathematics, Vol. 4. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611971316.

[11] Marcos Diez García. 2022. *FLE-UoE GitHub repository.* https://github.com/marcosdg/FLE-UoE Last accessed on 30 January 2022.

[12] Fred Glover, Bahram Alidaee, César Rego, and Gary Kochenberger. 2002. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research* 137, 2 (2002), 272–287. Elsevier. https://doi.org/10.1016/S0377-2217(01)00209-0.

[13] Fred Glover, Gary Kochenberger, and Yu Du. 2019. Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4OR–A Quarterly Journal of Operations Research* 17, 4 (2019), 335–371. Springer. https://doi.org/10.1007/s10288-019-00424-y.

[14] Nicolò Gusmeroli and Angelika Wiegele. 2021, in press. EXPEDIS: An exact penalty method over discrete sets. *Discrete Optimization* (2021, in press). Elsevier. https://doi.org/10.1016/j.disopt.2021.100622.

[15] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2, 5 (2014), 1–15. Frontiers. https://doi.org/10.3389/fphy.2014.00005.

[16] Satoshi Matsubara et al. 2020. Digital Annealer for High-Speed Solving of Combinatorial optimization Problems and Its Applications. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC).* IEEE, Beijing, China, 667–672. https://doi.org/10.1109/ASP-DAC47756.2020.9045100.

[17] Catherine C. McGeoch, Richard Harris, Steven P. Reinhardt, and Paul I. Bunyk. 2019. Practical Annealing-Based Quantum Computing. *Computer* 52, 6 (2019), 38–46. IEEE. https://doi.org/10.1109/MC.2019.2908836.

[18] Hiroshi Nakayama et al. 2021. *Third Generation Digital Annealer Technology.* Technical Report. Fujitsu Ltd.

[19] Gerhard Reinelt. 1991. TSPLIB—A Traveling Salesman Problem Library. *INFORMS Journal on Computing* 3, 4 (1991). Springer. https://doi.org/10.1287/ijoc.3.4.376.

[20] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos López de Prado. 2016. Solving the Optimal Trading Trajectory Problem Using a Quantum Annealer. *IEEE Journal of Selected Topics in Signal Processing* 10, 6 (2016), 1053–1060. IEEE. https://doi.org/10.1109/JSTSP.2016.2574703.

[21] Dong Ku Shin, Z. Gürdal, and O. H. Griffin Jr. 1990. A Penalty Approach for Nonlinear Optimization with Discrete Design Variables. *Engineering Optimization* 16, 1 (1990), 29–42. Taylor & Francis. https://doi.org/10.1080/03052159008941163.

[22] Amit Verma and Mark Lewis. 2020. Optimal quadratic reformulations of fourth degree pseudo-Boolean functions. *Optimization Letters* 14, 6 (2020), 1557–1569. Springer. https://doi.org/10.1007/s11590-019-01460-7.

[23] Amit Verma and Mark Lewis. 2020. Penalty and partitioning techniques to improve performance of QUBO solvers. *Discrete Optimization* (2020). Elsevier. https://doi.org/10.1016/j.disopt.2020.100594.