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

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]:

\[
q(x) = c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{1 \leq i < j \leq n} c_{ij} x_i x_j
\]

with fixed scalar coefficients \( c_0, c_i \) and \( c_{ij} \). 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 \( g(x) \geq 1 \) 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).

---

**Figure 1: Validity of penalty weights.**

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

Known techniques to derive such valid penalty weights are: (a) upper-bounding \( f_{\text{max}} \) (respectively, lower-bounding \( f_{\text{min}} \)) 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 suboptimal 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.
Exact and Sequential Penalty Weights in Quadratic Unconstrained Binary Optimisation with a Digital Annealer

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 weingl) 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 \( w_L \), 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 possiform-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

Algorithm 1 (Standard) Sequential Penalty Algorithm

1: \( w \leftarrow 1 \) \hspace{1cm} \text{\textit{initial penalty weight}}
2: \textbf{for each iteration do}
3: \hspace{1cm} minimise \( h(x) = f(x) + w \cdot g(x) \) with one DA run
4: \hspace{1cm} \textbf{if} solution found is feasible \textbf{then}
5: \hspace{1cm} \hspace{1cm} record solution found and \( w \)
6: \hspace{1cm} \hspace{1cm} \textbf{if} \( w \) \hspace{1cm} \hspace{1cm} \hspace{1cm} \textbf{record solution and solution found}
7: \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \textbf{return} minimum feasible solution found and corresponding \( w \)

Algorithm 2 Scaled-sequential Penalty Algorithm

\textbf{Input:} \( w_L \leftarrow \text{upper-bound of a valid penalty weight} \)
\textbf{Input:} \( t \), maximum number of iterations \((t > 1)\)
1: \( w \leftarrow 1 \) \hspace{1cm} \text{\textit{initial penalty weight}}
2: \text{scale_factor} \leftarrow w_L^{1/(t-1)} \hspace{1cm} \text{\textit{power scale midpoint}}
3: \textbf{for each iteration do}
4: \hspace{1cm} minimise \( h(x) = f(x) + w \cdot g(x) \) with one DA run
5: \hspace{1cm} \textbf{if} solution found is feasible \textbf{then}
6: \hspace{1cm} \hspace{1cm} record solution and \( w \)
7: \hspace{1cm} \hspace{1cm} \textbf{return} minimum feasible solution found and corresponding \( w \)
8: \hspace{1cm} \textbf{else}
9: \hspace{1cm} \hspace{1cm} \text{update interval endpoint}
10: \hspace{1cm} \hspace{1cm} \text{\textbf{update interval endpoint}}
11: \hspace{1cm} \text{\textbf{return} minimum feasible solution found and corresponding \( w \)}

Algorithm 3 Binary Search Penalty Algorithm

\textbf{Input:} \( w_L \leftarrow \text{upper-bound of a valid penalty weight} \)
1: \text{interval endpoint} \( a \leftarrow 1 \); \text{interval endpoint} \( b \leftarrow w_L \)
2: \textbf{for each iteration do}
3: \hspace{1cm} \text{power scale midpoint}
4: \hspace{1cm} minimise \( h(x) = f(x) + w \cdot g(x) \) with one DA run
5: \hspace{1cm} \textbf{if} solution found is feasible \textbf{then}
6: \hspace{1cm} \hspace{1cm} record solution found and \( w \)
7: \hspace{1cm} \hspace{1cm} \text{\textbf{update interval endpoint}}
8: \hspace{1cm} \text{\textbf{else}}
9: \hspace{1cm} \hspace{1cm} \text{update interval endpoint}
10: \text{\textbf{return} minimum feasible solution found and corresponding \( w \)}
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 instances. We will also consider using the sequential methods of updating penalty weights for improving performance.

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

<table>
<thead>
<tr>
<th>Problem Category</th>
<th>Instance Name</th>
<th>Best ARPD</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mincut</td>
<td>data</td>
<td>189</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>add20</td>
<td>189</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>Min-cut</td>
<td>20</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>add32</td>
<td>11</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>205.15</td>
<td>206.21</td>
</tr>
<tr>
<td>Mincut</td>
<td>Average</td>
<td>20.00</td>
<td></td>
</tr>
<tr>
<td>TSP</td>
<td>fri26</td>
<td>397</td>
<td>3.08</td>
</tr>
<tr>
<td></td>
<td>bays29</td>
<td>2020</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>dantzig42</td>
<td>699</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>brazil58</td>
<td>25395</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>st70</td>
<td>675</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.45</td>
<td>2.05</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Problem Category</th>
<th>Instance Name</th>
<th>Best ARPD</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>add20</td>
<td>596</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td>189</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>Min-cut</td>
<td>90</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>uk</td>
<td>20</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>add32</td>
<td>11</td>
<td>5.45</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>11.01</td>
<td>11.01</td>
</tr>
<tr>
<td>Mincut</td>
<td>Average</td>
<td>206.21</td>
<td></td>
</tr>
<tr>
<td>TSP</td>
<td>fri26</td>
<td>937</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>bays29</td>
<td>2020</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>dantzig42</td>
<td>699</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>brazil58</td>
<td>25395</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>st70</td>
<td>675</td>
<td>4.35</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>5.82</td>
<td>3.73</td>
</tr>
</tbody>
</table>

### REFERENCES


