Multi-objective QUBO Solver: Bi-objective Quadratic Assignment Problem

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

Manuel López-Ibáñez
The University of Manchester
Manchester, United Kingdom
manuel.lopez-ibanez@manchester.ac.uk

Richard Allmendinger
The University of Manchester
Manchester, United Kingdom
richard.allmendinger@manchester.ac.uk

Matthieu Parizy
Fujitsu Laboratories Ltd.
Kawasaki, Japan
parizy.matthieu@fujitsu.com

ABSTRACT
Quantum and quantum-inspired optimisation algorithms are designed to solve problems represented in binary, quadratic and unconstrained form. Combinatorial optimisation problems are therefore often formulated as Quadratic Unconstrained Binary Optimisation Problems (QUBO) to solve them with these algorithms. Moreover, these QUBO solvers are often implemented using specialised hardware to achieve enormous speedups, e.g., Fujitsu’s Digital Annealer (DA) and D-Wave’s Quantum Annealer. However, these are single-objective solvers, while many real-world problems feature multiple conflicting objectives. Thus, a common practice when using these QUBO solvers is to scalarise such multi-objective problems into a sequence of single-objective problems. Due to design trade-offs of these solvers, formulating each scalarisation may require more time than finding a local optimum. We present the first attempt to extend the algorithm supporting a commercial QUBO solver as a multi-objective solver that is not based on scalarisation. The proposed multi-objective DA algorithm is validated on the bi-objective Quadratic Assignment Problem. We observe that algorithm performance significantly depends on the archiving strategy adopted, and that combining DA with non-scalarisation methods to optimise multiple objectives outperforms the current scalarised version of the DA in terms of final solution quality.

CCS CONCEPTS
• Computer systems organization → Quantum computing;
• Mathematics of computing → Combinatorial optimization;
• Applied computing → Multi-criterion optimization and decision-making.

KEYWORDS
Multi-objective, QUBO, bi-objective QAP, Digital Annealer

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Mayowa Ayodele, Richard Allmendinger, Manuel López-Ibáñez, and Matthieu Parizy. 2022. Multi-objective QUBO Solver: Bi-objective Quadratic Assignment Problem. In Genetic and Evolutionary Computation Conference (GECCO ’22), July 9–13, 2022, Boston, MA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528698

1 INTRODUCTION
There has been significant research interest in quantum-inspired and quantum optimisation algorithms in recent years. Such algorithms are able to exploit the use of specialised hardware to solve optimisation problems much quicker than classical algorithms implemented on general purpose machines [1, 3].

Quantum-inspired as well as quantum algorithms are designed for solving problems that are in binary and quadratic form only. Quadratic Unconstrained Binary Optimisation (QUBO), also sometimes referred to as Unconstrained Binary Quadratic Programming (UBQP) or ising model, is one of the most widely used representations when solving combinatorial optimisation problems [31]. Knapsack, travelling salesman and graph partitioning problems can be formulated as QUBO problems [22].

Commercial QUBO solvers such as D-wave’s Quantum Annealer (QA) and Fujitsu’s Digital Annealer (DA) have evolved in the last ten years. Fujitsu’s first generation DA, introduced in 2018, was able to optimise QUBOs with up to 1,024 bits. The third generation DA is now able to optimise Binary Quadratic Problems (BQPs) with up to 100,000 bits [13]. BQPs include QUBO and other binary and quadratic formulations that may have constraints. D-wave One released in 2011 was able to optimise QUBOs with 128 quantum bits (qubits). D-wave’s Advantage released in 2020 contains at least 5,000 qubits [24]. D-Wave’s Advantage is quantum based [24] whereas DA is CMOS hardware based [3].

In addition to quantum annealing and digital annealing algorithms used in commercial solvers, other algorithms such as estimation of distribution algorithms, scatter search, simulated annealing (SA), tabu search and genetic algorithm have also been applied to QUBOs [5, 15].

Multi-objective optimisation has been a topic of interest for many years in operations research [32]. There are various approaches of solving multi-objective problems; a priori (e.g. scalarisation, goal programming, ε-constraint), a posteriori, and interactive [8, 9]. Although a posteriori algorithms, based on Pareto optimality, have been designed to solve QUBOs with multiple objectives [11, 17],
a priori methods are used when solving multi-objective problems with the QA or DA since they are single-objective solvers. For example, scalarisation was used to solve a bi-objective listing optimisation problem with the QA [25] and to solve the bi-objective problem of routing and wavelength assignment with the DA [29]. The \(\epsilon\)-constraint approach was used with the QA to solve the portfolio optimisation problem [28].

In this study, we propose a new multi-objective DA (MDA) algorithm for Pareto optimisation by redesigning the existing DA algorithm, which is based on local search, to handle multiple objectives and an archive of solutions. The first step to extending single-objective local search to multi-objective optimisation is to update the acceptance criteria [26]. Since the DA is similar to SA, we update the acceptance criteria \[26\]. Since the DA is similar to SA, we update the acceptance criteria. Different methods of updating the archive are also compared. Finally, we compare a scalarisation-based single-objective DA (SB-DA) with MDA. Due to the peculiarities of the problem of routing and wavelength assignment with the DA [29].

As a benchmark problem, we consider the bi-objective Quadratic Assignment Problem (QAP). The DA has been shown to present competitive performance on single-objective QAP instances [23, 30]. In [23], the DA was able to solve QAPLIB [7] instances to optimality up to 165,000 times faster than CPLEX.

The rest of this paper is structured as follows. A formal description as well as the QUBO formulation of the bi-objective QAP is presented in Section 2. Details of the proposed algorithm for multi-objective optimisation within DA are presented in Section 3. Problem instances, parameter settings as well as performance measure used in this study are described in Section 4. Results are analysed in Section 5. Finally, conclusions are presented in Section 6.

## 2 PROBLEM DESCRIPTION

In this section, the bi-objective problem is formally defined, and the QUBO formulation is presented.

### 2.1 Single-Objective Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) [16] can be described as the problem of assigning a set of \(n\) facilities to a set of \(n\) locations. For each pair of locations, a distance is specified. For each pair of facilities, a flow (or weight) is specified. The aim is to assign each facility to a unique location such that the sum of the products between flows and distances is minimised.

Formally, an instance of the QAP consists of two \(n \times n\) input matrices \(H = [h_{ij}]\) and \(D = [d_{u\sigma}]\), where \(h_{ij}\) is the flow between facilities \(i\) and \(j\), and \(d_{u\sigma}\) is the distance between locations \(u\) and \(\sigma\). A solution to the QAP is a permutation \(\sigma = (\sigma_1, \ldots, \sigma_n)\) where \(\sigma_i\) represents the location assigned to facility \(i\). The cost function is formally defined as follows

\[
\text{minimise } f(\sigma) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} \cdot d_{\sigma_i \sigma_j} .
\]

### 2.2 QUBO formulation of the QAP

QUBO problems are unconstrained, quadratic and of binary form generally defined as follows:

\[
E(x) = x^T Q x + q , \tag{2}
\]

where \(Q\) represents a \(m \times m\) matrix, \(q\) is a constant term, a solution \(x = (x_1, \ldots, x_m)\), \(x_i \in \{0, 1\}\), is an \(m\)-dimensional binary vector, and \(E(x)\) is the energy (or fitness) of \(x\).

The QAP can be formulated as QUBO using Eq. (3) where the cost function \(c(x)\) and the constraint function \(g(x)\) are presented in Eqs. (4) and (5), respectively. The penalty weight is denoted by \(\alpha\) and set using the method presented later in Section 3.2 (Eq. (16)).

\[
c(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} d_{u\sigma_i \sigma_j} x_{ij} \tag{4}
\]

\[
g(x) = \sum_{i=1}^{n} \left( 1 - \sum_{u=1}^{n} x_{iu} \right)^2 + \frac{n}{\alpha} \sum_{u=1}^{n} \left( 1 - \sum_{i=1}^{n} x_{iu} \right)^2 \tag{5}
\]

In Eq. (1), the fitness function \(f(\sigma)\) is equivalent to cost function \(c(x)\) in (3), where \(\sigma\) is a solution represented as a permutation and \(x\) is the equivalent two-way one-hot (permutation matrix) solution. The two-way one-hot representation is often used to encode permutation problems when transforming them to QUBO [18, 22]. In this representation, each entity with index \(i\) (e.g. location \(i\) in the QAP) is represented by a substring of \(n\) zeros but the \(i^{th}\) bit is set to 1, thus a binary variable \(x_{iu}\) indicates whether facility \(u\) is assigned to location \(i\) or not. To prevent solutions that cannot be decoded to a valid permutation, \(g(x)\) penalises any solution where each row and/or column does not sum to one (Eq. 5). The function \(g(x)\) is non-negative: either \(g(x) = 0\) if the solution \(x\) can be decoded to a valid permutation or \(g(x) > 0\) for infeasible solutions. The value of \(g(x)\) increases according to the degree of constraint violation.

An example of a valid permutation and its encoding is

\[
\sigma = (3,1,2) \Leftrightarrow x = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6}
\]

Typically \(x\) is represented in vector format, where \(x\) in Eq. (6) is the same as

\[
x = (0, 0, 1, 1, 0, 0, 0, 1, 0) , \tag{7}
\]

meaning that QUBO solutions have dimension \(m = n^2\).

As an example, consider the following QAP instance:

\[
H = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 3 & 4 \\ 3 & 0 & 6 \\ 4 & 6 & 0 \end{bmatrix} , \tag{8}
\]
for which the corresponding QUBO matrix $C$ and the constant term $p_c$ representing the cost function are presented as follows:

$$C = \begin{bmatrix}
0 & 0 & 0 & 0 & 6 & 8 & 0 & 12 & 16 \\
0 & 0 & 0 & 6 & 0 & 12 & 0 & 24 & 0 \\
0 & 0 & 0 & 8 & 12 & 0 & 16 & 24 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 12 \\
0 & 0 & 0 & 0 & 0 & 8 & 12 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$p_c = 0$

and the QUBO matrix $G$ and the constant term $p_g$ representing the constraint function are presented as follows:

$$G = \begin{bmatrix}
-2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & -2 & 2 & 0 & 2 & 0 & 2 & 0 & 0 \\
0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 2 & 2 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 2 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 2 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2
\end{bmatrix}$$

$p_g = 6$

An example of a valid solution (i.e., valid permutation) is

$$\sigma = [3, 2, 1, 4, 1, 0, 0, 0, 1, 0]$$

$$f(\sigma) = 0 \cdot 0 + 1 \cdot 4 + 2 \cdot 6 + 1 \cdot 4 + 0 \cdot 0 + 1 \cdot 3 + 2 \cdot 6 + 1 \cdot 3 + 0 \cdot 0 = 38$$

$$c(x) = x^T C x + p_c = 38 + 0 = 38$$

$$g(x) = x^T G x + p_g = -6 + 6 = 0$$

while an invalid solution would be

$$\sigma = [3, 2, 1, 4, 1, 0, 0, 0, 1, 0]$$

$$f(\sigma) = 0 \cdot 0 + 1 \cdot 4 + 2 \cdot 6 + 1 \cdot 4 + 0 \cdot 0 + 1 \cdot 3 + 2 \cdot 6 + 1 \cdot 3 + 0 \cdot 0 = 36$$

$$c(x) = x^T C x + p_c = 36 + 0 = 36$$

$$g(x) = x^T G x + p_g = -4 + 6 = 2$$

In both cases $c(x) \equiv f(\sigma)$.

### 2.3 Bi-Objective Quadratic Assignment Problem

The multi-objective QAP has real-world relevance in scenarios like the hospital layout problem, where it may be important to simultaneously minimise the flows of doctors, patients, hospital guests, etc [14]. In this study, we consider the bi-objective QAP.

An instance of the bi-objective QAP consists of a $2 \times n \times n$ flow matrix $H = [h_{kij}]$ and a $n \times n$ distance matrix $D = [d_{uv}]$. The cost function of the bi-objective QAP is defined as

$$\min \tilde{f}(\sigma) = (f_1(\sigma), f_2(\sigma))$$

where $f_k(\sigma) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{kij} \cdot d_{ui} \cdot d_{vj}, \ k \in \{1, 2\}$. (9)

The equivalent QUBO formulation of $f_k(\sigma)$ is

$$c_k(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{u=1}^{n} \sum_{v=1}^{n} h_{kij} \cdot d_{ui} \cdot x_{iu} \cdot x_{jv}, \ k \in \{1, 2\}.$$ (10)

When a two-way one-hot solution $x$ is equivalent to a permutation $\sigma$, then $f_k(\sigma) \equiv c_k(x)$. The constraint function $g(x)$ (Eq. 11) remains the same as that of the single-objective formulation (Eq. 5).

We explore solving the QUBO formulation of the bi-objective QAP using a single-objective algorithm and a multi-objective algorithm. In the single-objective solution approach, the weighted sum method is used as follows

$$\min E(x) = \gamma \cdot c_1(x) + (1-\gamma) \cdot c_2(x) + \alpha \cdot g(x), \ \gamma \in [0, 1].$$ (11)

The formulation for the multi-objective (Pareto optimisation) solution approach is.

$$\min \tilde{E}(x) = (c_1(x) + \alpha_1 \cdot g(x), c_2(x) + \alpha_2 \cdot g(x))$$ (12)

### 3 DIGITAL ANNEALER

In this section, we present the SB-DA as well as the proposed MDA algorithm. The algorithm that supports the first generation DA [3] is used in this study. In particular, we use a CPU implementation of Alg. 2 and Alg. 3 to solve the bi-objective QAP when formulated as a single-objective problem and multi-objective problem, respectively.

#### 3.1 Digital Annealer Algorithm

In Alg. 1, we set the QUBO matrix $Q$ to be solved, the maximum number of iterations allowed ($I_{\text{max}}$), initial temperature ($\delta_0$), final temperature ($\delta_f$), temperature decay ($\xi$) and the offset increase rate

**Algorithm 1 DA Algorithm**

Require: $Q$, $I_{\text{max}}$, $\delta_0$, $\delta_f$, $\xi$, $\beta$

1: $x \leftarrow$ a random binary solution of length $m$

2: $\delta \leftarrow \delta_0$

3: for iteration = 1 to $I_{\text{max}}$ do

4: if $\delta \leq \delta_f$ then $\delta \leftarrow \delta \cdot (1 - \xi)$

5: calculate $\Delta E$, energy difference between $x$ and $N(x)$

6: for each variable $i$ in $x$ do in parallel

7: propose a flip using $\Delta E_i - E_{\text{offset}}$

8: if acceptance criteria is satisfied then record flip

9: end for

10: if at least one flip meets the acceptance criteria then

11: choose one flip from random from recorded flips

12: set $x$ as solution corresponding to the selected flip

13: $E_{\text{offset}} \leftarrow 0$

14: else $E_{\text{offset}} \leftarrow E_{\text{offset}} + \beta$

15: end if

16: end for

17: return best solution found

469
parameter (\(\beta\)), which is used to escape local optima. We use a simple cooling scheme presented in line 4 to reduce the temperature from \(\delta_0\) to \(\delta_f\).

The DA architecture is designed to run on Fujitsu’s specialised hardware [23] (in a similar manner that some algorithms are designed to run on Graphical Processing Units), such that all neighbouring solutions are explored in parallel and in constant time regardless of the number of neighbours. This approach significantly improves acceptance probabilities when compared to the classic SA algorithm [3]. The DA algorithm does not completely evaluate a solution, rather, it calculates the energy difference between any solution \(x\) and all its \(m\) neighbours, \(N(x)\). In the CPU implementation used in this study, we calculate the energy difference (\(\Delta E\)) between \(x\) and all its neighbours \(N(x)\) as

\[
\Delta E = \left\{ P_{ji} \cdot (1 - 2x_i) + \sum_{j=1}^{m} P_{ji} \cdot (1 - 2x_j) \cdot x_i, \quad \forall i \in [1, m] \right\}
\]

\[
P = Q + Q^T, \quad P_{ji} = P_{ij} - Q_{ii} \forall i \in [1, m]
\]

(13)

where \(Q\) and \(m\) are used to denote the QUBO matrix and its size, respectively. \(P\) is the sum of \(Q\) and the transpose \(Q\) without the linear terms. \(P\) only needs to be computed once, regardless of the number of iterations or runs.

To estimate \(\Delta E\) (Eq. 13), the decision to add or subtract the linear terms (i.e. \(i = j\)) is made depending on the variable \(x_i\) being a 0 or 1. The quadratic terms (i.e. \(i \neq j\)) are deducted only if \(x_i = 1\). With this simple method, we can estimate \(\Delta E\) for all neighbours of \(x\) in one go. The energy (constrained objective function) of a solution \(y\) that is the \(i^{th}\) neighbour of \(x\) is therefore \(E(y) = E(x) + \Delta E_i\).

A new solution \(y\) is accepted if its \(\Delta E\) value is negative (i.e. there is an improvement in the energy). It can also be accepted if \(y\) is worse but within the acceptance probability. The acceptance probability is calculated as

\[
Pr(y \text{ is accepted}) = \exp(-\frac{\Delta E_i - E_{\text{offset}}(x)}{\delta})
\]

(14)

The probability of accepting a solution \(y\), which is the \(i^{th}\) neighbour of a current solution \(x\), depends on the quality of the solution compared to \(x\) (\(\Delta E_i\)), how long the search has been trapped in local optima (\(E_{\text{offset}}(x)\)) and the current temperature \(\delta\). Higher temperatures mean more exploration and higher probability of accepting degrading solutions (larger positive \(\Delta E_i\) values). \(E_{\text{offset}}\) is set to 0 when the algorithm is not trapped, but increases by \(\beta\) for each iteration that the algorithm is not able to escape a local optimum.

### 3.2 Scalarisation-Based DA Algorithm

The SB-DA algorithm is described in Alg. 2. In addition to parameters \((I_{\text{max}}, \delta_0, \delta_f, \xi, \beta, \Gamma)\) used by the DA, SB-DA requires additional inputs, which are \(R, S, G\) and \(\Gamma\). QUBO matrices \(R, S\) and \(G\) represent the first objective, second objective and constraint functions, respectively, while \(\Gamma\) is a set of scalarisation weights.

In line 3 of Alg. 2, \(C\) is the QUBO matrix that represents the aggregated cost function. Penalty weight \(\alpha\) (line 4) is set using method presented in Eq. (16).

### Algorithm 2 SB-DA Algorithm

Require: \(R, S, G, I_{\text{max}}, \delta_0, \delta_f, \xi, \beta, \Gamma\)

1. \(A \leftarrow \emptyset\)  \text{ » Initialise archive}
2. \textbf{for each } \(y \in \Gamma \text{ do}
3. \quad C \leftarrow y \times R + (1 - y) \times S
4. \quad \text{ estimate } \alpha \text{ from } C
5. \quad Q \leftarrow C + \alpha \times G
6. \quad \text{ execute the DA with parameters } Q, I_{\text{max}}, \delta_0, \delta_f, \xi, \beta
7. \quad \text{ add } x \text{ to } A
8. \textbf{end for}
9. \textbf{return} all non-dominated solutions from archive \(A\)

\[
w = \max \left\{ -C_{ii} - \sum_{j=1}^{m} \min\{C_{ij}, 0\} \forall i \in [1, m], C_{ii} + \sum_{j=1}^{m} \max\{C_{ij}, 0\} \forall i \in [1, m] \right\}
\]

(15)

\[
\alpha = \frac{w}{2}
\]

(16)

Equation (15) was originally proposed in [31], where \(w\) is an upper bound of the difference in \(C\) that can be achieved by either turning a bit off (from 1 to 0) or on (from 0 to 1). Verma and Lewis [31] suggested that \(w\) can be further reduced using information from the constraint function. Equation (16) was proposed in [4] for permutation problems represented as two-way one-hot, given that the minimum constraint function value of any infeasible solution (i.e. solutions that are one flip away from feasibility) is 2.

The SB-DA algorithm makes multiple calls to the DA (Alg. 1) using different \(Q\) matrices (\(Q\) is a weighted aggregate of the QUBOs representing the objectives and constraints) and returns a set of non-dominated solutions.

### 3.3 Multi-objective Digital Annealer Algorithm

In this study, we adapt the DA algorithm as a multi-objective algorithm where the energies corresponding to all the objective functions \((E_1(x), E_2(x))\) are simultaneously optimised. The relationship between any two solutions \(x\) and \(y\) can be defined in any of the following ways.

1. \(x\) is equivalent to \(y\) (denoted by \(x \sim y\)), if \(E_i(x) = E_i(y) \forall i \in \{1, 2\}\)
2. \(x\) dominates \(y\) (denoted by \(x \prec y\)), if \(E_i(x) \leq E_i(y) \forall i \in \{1, 2\}\) and \((x \sim y)\)
3. \(x\) and \(y\) are mutually non-dominated (denoted by \(x \parallel y\)), if \(\exists i, j \in \{1, 2\}\) such that \(E_i(x) < E_i(y)\) and \(E_j(x) > E_j(y)\)

The MDA algorithm proposed in this work is presented in Alg. 3. Similar to the DA, MDA uses temperature parameters \(\delta_0, \delta_f\) and \(\xi\) as well as the stopping criteria \((I_{\text{max}})\). Additional inputs used are QUBO matrices \(Y\) and \(Z\) as well as archive size \(s\). QUBO matrices \(Y\) and \(Z\) are derived from \(R\) and \(S\) and \(G\) that are QUBO matrices representing the first objective, second objective and constraint.
Algorithm 3 MDA Algorithm

Require: $Y$, $Z$, $\Delta t$, $\delta_0$, $\delta_f$, $\xi$, $s$

1. Initialise archive $A$ with maximum size $s$
2. $x \leftarrow$ a random binary solution of length $m$
3. $\delta \leftarrow \delta_0$
4. for iteration $= 1$ to $\Delta t$ do
5.   if $\delta > \delta_f$ then $\delta \leftarrow \delta - (1 - \xi)$
6.   calculate $\Delta E$, energy difference between $x$ and $N(x)$ for all objectives
7.   for each variable $i$ in $x$ do in parallel
8.     propose a flip using $\Delta E_{ik}$, $\forall k \in \{1, 2\}$
9.     if acceptance criteria is satisfied then record flip
10. end for
11. if at least one flip meets the acceptance criteria then
12.   choose one flip at random from recorded flips
13.   set $x$ as solution corresponding to the selected flip
14. end if
15. else
16.   $x \leftarrow$ a randomly selected solution from $A$
17. end if
18. end for
19. return all non-dominated solutions from archive $A$

functions respectively (Eq. 17):

$$Y = R + \alpha_1 \times G, \quad Z = S + \alpha_2 \times G$$

Penalty weights $\alpha_1$ or $\alpha_2$ are derived by substituting $R$ or $S$ for $C$ in Eq. (15).

Like the DA, neighbours $N(x)$ of a solution $x$ are evaluated by exploring the difference in energy $\Delta E$. In this case however, the difference is computed using QUBOs ($Y$ and $Z$, Eq. 17) representing each objective. $\Delta E_{ik}$ (line 8) denotes the energy difference between $x$ and the $i$th neighbour of $x$ w.r.t. the $k$th objective. $\Delta E_{i1}$ (or $\Delta E_{i2}$) is estimated by substituting $Y$ (or $Z$) for $Q$ in Eq. (13).

The difference between the SB-DA algorithm used in this study and the proposed MDA are: (1) SB-DA solves a sequence of QUBOs independently while the MDA solves two QUBOs simultaneously, they therefore require different acceptance criteria, (2) multiple pre-processing of QUBOs are required in the SB-DA, once for each scalarisation weight in $\Gamma$, while pre-processing is done only once for MDA and (3) the methods of escaping local optima are different.

Acceptance Criteria. Two acceptance criteria (strict and lenient), adapted from [2], are used in this study. Given that $y$ is the $i$th neighbour of $x$, the strict acceptance criteria (Eq. 18) uses the product of the probabilities of accepting a solution $y$ within the context of each individual objective. The lenient criteria (Eq. 19) uses the highest probability of accepting a solution $y$ with respect to any of the objectives.

$$\Pr(y \text{ is accepted}) = \prod_{k=1}^{2} \Pr^k(y \text{ is accepted})$$

$$\Pr(y \text{ is accepted}) = \max_{k=1}^{2} \Pr^k(y \text{ is accepted})$$

$$\Pr^k(y \text{ is accepted}) = \exp\left(\min\{0, -\Delta E_{ik}/\delta\}\right).$$

The lenient acceptance rule provides the certainty that non-comparable or dominating solutions are accepted while the strict acceptance rule allows for deeper exploration [2].

Escaping Local Optimal. The parameter $\beta$, used to escape local optimal in the DA, is not used in the MDA because the latter is able to explore other areas of the search space by switching to another solution in the archive (see line 16).

Updating the Archive. In the MDA, an archive is used to store multiple non-dominated solutions. A fixed-size archive of size $s$ is used in this study. At each iteration, one of the neighbouring solutions that meet the acceptance criteria is randomly selected as a potential solution to be added to the archive (line 12). We explore two methods of updating the archive: A more explorative and a more exploitative method.

Algorithm 4 update($A, x$): Explorative Method

Require: $A$: archive, $x$: new solution, $s$: maximum size

1. dominates $\leftarrow$ FALSE
2. for each $y \in A$ do
3.   if $x < y$ then
4.     dominates $\leftarrow$ TRUE
5.     break
6. end if
7. end for
8. if dominates then
9.   replace $y$ with $x$ in $A$
10. else if $|A| < s$ then
11.   add $x$ to $A$
12. end if

In the explorative method (Alg. 4), if an accepted solution $x$ dominates any solution in the archive, then $x$ replaces the first of the dominated solutions. However, if the accepted solution $x$ does not dominate any solution but the archive is not full, then such solution $x$ is also added to the archive.

Algorithm 5 update($A, x$): Exploitative Method

Require: $A$: archive, $x$: new solution, $s$: maximum size

1. $B \leftarrow \{y \in A \mid x < y\}$
2. if $\exists z \in A$, $z < x$ then
3.   if $|B| > 0$ then
4.     select a solution $y$ from $B$ at random
5.     replace $y$ with $x$ in $A$
6.   else if $|A| < s$ then
7.     add $x$ to $A$
8. end if
9. end if

While the explorative method updates the archive with solutions that meet the acceptance criteria, such solutions must be non-dominated for it to be added to the archive in the more exploitative method (Alg. 5). The non-dominated solution $x$ is added to the archive either by replacing a dominated solution selected at random or by filling a new position in the archive (if the size of the archive is less than the maximum size $s$).
4 EXPERIMENTAL SETTINGS

In this section, the instances used in this study are described. The parameter settings and performance measures are also defined.

4.1 Bi-Objective QAP Instances

We use publicly available bi-objective QAP instances [20, 27]. The benchmark consists of instances classed as structured or unstructured, with 25, 50 or 75 locations (and facilities) and different levels of correlations between their objectives. It also consists of best known objectives (relating to the Pareto sets) for the problem sets. Details of how the problems are generated can be found in [14].

In this study, we use structured instances with varying numbers of locations/facilities $n \in \{25, 50\}$ and different levels of correlations $\rho \in \{-0.75, 0, 0.75\}$ between the objectives. Instances are named as qapStr.$n.X.i$, where $n$ is the instance size, $X \in \{n75, 0, p75\}$ denotes the correlation, and $i = (1, 2, 3)$ is a numerical identifier to distinguish instances of the same size and correlation. Instances that have negative correlation between their objectives ($\rho = -0.75$) simulate problems that have conflicting objectives. Instances with no correlation between the objectives ($\rho = 0$) simulate scenarios where the objectives are independent of each other. Where there is positive correlation ($\rho = 0.75$), we expect that minimising one objective should often lead to the minimisation of the other objective.

4.2 Parameter Settings

The parameter settings used in this study for SB-DA and MDA are presented in Table 1. These values were chosen based on preliminary experiments. Each algorithm configuration was run on all problem instances, each run was independently repeated 20 times with unique random seeds.

Furthermore, to prevent bias towards one of the objectives, we consider normalisation. We set the largest QUBO coefficient representing any objective QUBO $R$ and $S$ to $2^{23}$. These are inputs in SB-DA (Alg. 2) and used to compute $Y$ and $Z$ (Eq. 17), which are inputs in the MDA (Alg. 3). The maximum value ($2^{23}$) was derived empirically as we found that it was large enough to capture the information in the original QUBO. This was true for all the instances used in this study. We compare the quality of solutions returned when normalised or non-normalised QUBOs are used (Section 5.1).

We compare results for SB-DA and MDA using non-dominated solutions from all 20 runs.

4.3 Performance Criteria

Several performance measures have been proposed for comparing multi-objective optimisers. Following best practices [33], we use the Empirical Attainment Function (EAF) and the hypervolume metric in this study.

The EAF of an algorithm gives the probability, estimated from multiple runs, that the non-dominated set produced by a single run of the algorithm dominates a particular point in the objective space. The visualisation of the EAF [12] has been shown as a suitable graphical interpretation of the quality of the outcomes returned by local search methods. The visualisation of the differences between the EAFs of two alternative algorithms indicates how much better one method is compared to another in a particular region of the objective space [21]. The EAF visualisations were done using the eaf R package.²

The hypervolume quality metric measures the area in the objective space that is dominated by at least one of the points of a non-dominated set and bounded by a given reference point that must be dominated by all points under comparison. Larger hypervolume values indicate that the non-dominated solutions are closer to the Pareto front. In this study, we calculate the hypervolume using the pymoo Python library [6]. Before computing the hypervolume, we normalise the energies of all the solutions generated in the study (together with the known energies for the Pareto optimal solutions) to the range $[1, 2]$. The reference point is set as $(2.1, 2.1)$, hence, the maximum hypervolume possible is 1.21.

5 RESULTS

In this section, we first assess the effect of normalisation in the algorithm. We also compare the two acceptance criteria (lenient vs strict) and archiving methods (explore vs exploit) in MDA. Finally, we compare the quality of results produced by the SB-DA with those produced by MDA.

5.1 Effect of QUBO Normalisation

Figures 1 and 2 present the hypervolume of the non-dominated solutions across 20 runs of the SB-DA and MDA respectively. We use the paired $t$-test to test for statistical significance (the corresponding p-values are shown in the plot). While Figures 1 and 2 only show results for the first instance of each set in qapStr.50 (details of problems set are presented in Section 4.1), results for all nine instances consisting of 25 locations and facilities (qapStr.25) as well as all nine instances consisting of 50 locations and facilities (qapStr.50) are presented in the supplementary material.

In SB-DA, the normalised QUBOs led to significantly better (p-value $\leq 0.05$) hypervolume values on all nine qapStr.25. We also show similar results on qapStr.50 instances with no correlation (qapStr.50.0) or positive correlation (qapStr.50.75) between the objectives. However, for instances with negative correlation between the objectives (qapStr.50.75), the SB-DA with normalised QUBOs did not present significantly better results on two of three instances when compared to using QUBOs that are not normalised.

| Table 1: SB-DA and MDA parameters and their settings. |
|-----------------------------------------------------|
| Parameter | SB-DA values | MDA values |
|-----------|--------------|------------|
| Initial temperature $\delta_0$ | $10^9$ | $10^9$ |
| Final temperature $\delta_f$ | $10^5$ | $10^5$ |
| Number of iterations $l_{max}$ | $0.25 \cdot m^2$ | $0.25 \cdot m^2$ |
| Number of runs | 20 | 20 |
| Offset increase rate $\beta$ | $\delta_0 / (0.25 \cdot m^2)$ | N/A |
| Temperature decay $\xi$ | 0.001 | 0.001 |
| Scalarisation weights $\Gamma$ | $\{0.0, 0.1, 0.2, \ldots, 1.0\}$ | N/A |
| Size of archive $s$ | N/A | $m$ |

²http://lopez-ibanez.eu/eaf/tools
We use the MDA with more explorative archiving method and strict acceptance criteria in this section because these are the best settings for the MDA (Section 5.2). In the MDA, there are no statistically significant (p-value > 0.05) differences between using the MDA with QUBOs that are either normalised or not (Figure 2).

For consistency, results presented in the following sections relating to SB-DA and MDA use normalised QUBOs.

**5.2 MDA: Assessing Algorithm Design Choices**

We compare the more explorative archive update method with the more exploitative archive update method using different acceptance criteria (strict and lenient). We test for significance in the difference between mean hypervolume values (20 independent runs) using the Mann-Whitney-Wilcoxon two-sided test with Bonferroni correction. We show the corresponding p-values in Figure 3.

Figure 3 shows that the MDA using a more explorative method of updating the archive presented better results (larger hypervolume) than the more exploitative method. The difference between the performance (hypervolume) of the MDA with strict and lenient criteria within the context of the more explorative method (Explore Lenient vs Explore Strict) are not of statistical significance (p-value > 0.05).

Using the exploitative method of updating the archive (Exploit Strict and Exploit Lenient) leads to larger variance between the hypervolume across all 20 runs compared to the exploitative method.

In Figure 4, we examine the acceptance criteria (with the more explorative archiving method) in more detail looking at the hypervolume recorded at each iteration. We show that the strict criteria converges faster than the lenient criteria on qapStr.50.0.1. Similar results are observed on all instances (qapStr.25 and qapStr.50), results are presented in the supplementary material.

**5.3 Comparing SB-DA and MDA**

To understand the possible advantage of extending the DA as a multi-objective algorithm, we compare the SB-DA with the proposed MDA (using more exploratory archive method and the strict acceptance criteria).
GECCO ’22, July 9–13, 2022, Boston, MA, USA

Ayodele et al.

Figure 4: Comparing archive acceptance criteria in the MDA

Table 2: Number of non-dominated solutions in the archive (mean ± std. dev.).

|            | qapStr.25. | qapStr.50. |
|------------|------------|------------|
|            | 0.1 n75.1  | p75.1      |
| SB-DA      | 4 ± 1      | 5 ± 1      |
| MDA        | 36 ± 9     | 44 ± 8     |

Table 3: Computation Times in Seconds (mean ± std. dev.).

|            | qapStr.25. | qapStr.50. |
|------------|------------|------------|
|            | 0.1 n75.1  | p75.1      |
| SB-DA      | 47 ± 0     | 51 ± 1     |
| MDA        | 8 ± 1      | 8 ± 1      |

Figure 5 shows the location of the EAF differences between the outcomes of the SB-DA and MDA with respect to their corresponding EAFs. The darker regions are the locations where the corresponding algorithm is better than the other algorithm. We see that the MDA is able to reach better non-dominated solutions (shown by the darker regions) across the front, than the SB-DA. Larger darker regions are shown in the nine qapStr.50 instances compared to the qapStr.25 instances (results are shown in the supplementary material). There is no region in the objective space where the SB-DA outperforms the MDA on all eighteen instances.

We also compare the number of non-dominated solutions returned by the SB-DA and MDA in Table 2. The SB-DA returned 3-5 non-dominated solutions while the MDA returned 24-98 non-dominated solutions. This is not unexpected as the maximum number non-dominated solutions that SB-DA can return in one run is the size of the scalarisation weights (Γ) while MDA can return up to s = m solutions.

We further compared the speed of arriving at the non-dominated solutions returned by the SB-DA and MDA in Table 3. All executions were performed on a machine equipped with Ubuntu 18.04, Intel Xeon Gold 5218 CPU @ 2.30GHz, and 192GB Memory. Results show that the MDA is an average of 4-6 times quicker than SB-DA. We conclude that MDA can present better quality and larger quantity of non-dominated solutions faster than SB-DA. We note that the DA that uses specialised hardware is orders of magnitude faster than the CPU implementation used in this study.

6 CONCLUSION

In this study, we presented methods of solving the bi-objective QAP formulated as QUBO. We explored scalarisation as well as Pareto optimisation. In the scalarisation approach, we assessed the effect of QUBO normalisation such that QUBO representing different objectives are put on the same scale before they are aggregated. We use the existing DA algorithm, which is similar to SA, to solve aggregated QUBOs as a single-objective problem. Furthermore, we also explored methods of extending the existing single-objective DA to a multi-objective algorithm. We particularly focused on methods of updating the archive as well as solution acceptance criteria in the MDA. We show that more promising non-dominated solutions can be attained using the MDA compared to running the single-objective DA with multiple scalarisation weights. We also show that these more promising solutions can be reached by the MDA in significantly shorter time. Moreover, uploading multiple large QUBOs may become expensive when the DA implemented on specialised hardware is used due to the time needed to upload such QUBOs to the DA.

This work has multiple future directions. First, we plan to assess the sensitivity of parameters of the algorithms. Second, we plan to investigate better ways of aggregating QUBOs for multiple objectives. Third, we want to assess the performance of the proposed MDA on more problems. Finally, we wish to extend the proposed MDA for solving more than two objectives as well as explore other ways of handling exploration and archiving [10, 19].

474
