QuCOOP: A Versatile Framework for Solving Composite and Binary-Parametrised Problems on Quantum Annealers

Supplementary Material

This supplementary material provides a deeper analysis of the proposed optimisation framework and more experimental details and results. It includes the following sections:

- Proofs of Lemmas 1 to 3 claimed in Sec. 4 of the main text (Appendix A);
- Details of the derivation of QuCOOP's QUBO problem (9) of the main text (Appendix B);
- A technical comparison between our QuCOOP method and the Q-Match algorithm in solving permutation problems, as mentioned in Secs. 7.1 and 7.2 of the main text (Appendix C);
- Details on the QABLIB results in Sec. 7.1 of the main text, as well as an investigation of the performance of an iterative local search variant of QuCOOPon QAPLIB problems (Appendix D);
- Further experiments on shape matching from Sec. 7.2 of the main text, including failure cases on FAUST and results on the TOSCA dataset (Appendix E);
- Qualitative results of point set registration computed on D-Wave complementing the numerical results of Sec. 7.4 of the main text (Appendix F);
- Description of the annealing and embedding process on D-Wave's quantum annealers, relevant for the results in Sec. 7.4 of the main text (Appendix G).

The notation in this supplement mostly follows the conventions of the main text. We summarise them in Tab. I.

Symbol	Meaning
$\mathcal{S} \subsetneq \mathbb{R}^n$	Feasible domain of an optimisation problem of interest
$\mathcal{X} = \mathbb{B}^k$	Parameter set for the feasible domain $\mathcal{S}$
g^t	Linear Taylor approximation of g around iterate x^t
f^t	Local model, quadratic in g^t
∇g	Gradient of g with respect to x
$\langle \cdot, \cdot \rangle$	Standard inner product in $\mathbb{R}^k$

Table I. Main notations used in this paper and their meaning.

A. Proofs of the Lemmas

Proof of Lemma 1. We show that the series of objective function values computed by QuCOOP is non-increasing in either of the following cases:

1. **With no reparametrisation step and under following assumptions:**
   i) The constraint $g^t(x) \in \mathcal{S}$ is enforced with a penalty term of the form $\alpha^t h(x, x^t)$ with sufficiently large penalty factor $\alpha^t \in \mathbb{R}_+$ and a function h such that $h(x^t, x^t) = 0$, and $h(x, x^t) > 0$ for all $x \neq x^t$.
   ii) At points $g^t(x)$ in $\mathcal{S}$ that the algorithm can reach in a specific iteration, the linearised parametrisation g^t coincides with g, so that $g^t(x^{t+1}) = g(x^{t+1})$.
2. **Alternatively with the reparametrisation step**:
   It is always possible to change Algorithm 1 to a non-increasing algorithm by including, as mentioned in the main text, a step $x^{t+1} = g^{-1}(g^t(\tilde{x}^{t+1}))$, where $\tilde{x}$ is the output of Problem (8).

Proof. 1. **With no reparametrisation step:**
Using Assumptions i): Majorisation-Minimisation argument:
We want to show that $f(g(x^{t+1})) \leq f(g(x^t))$ for all t. According to the assumption in i) the Problem (8) can be explicitly written without constraint as

$$x^{t+1} \leftarrow \underset{x \in \mathbb{B}}{\arg\min}\ f^t(x), \qquad f^t(x) := f(g^t(x)) + \alpha^t h(x, x^t), \tag{22}$$

where $\alpha^t \in \mathbb{R}_{>0}$ is a penalty factor and h some function in x ensuring the feasibility so that $g^t(x) \in \mathcal{S}$. At iteration t, the QUBO solver finds the minimiser x^{t+1} of f^t among all $x \in \mathcal{X}$ with $g^t(x) \in \mathcal{S}$. Automatically, we have $f^t(x^{t+1}) \leq f^t(x^t) = f(g(x^t))$. It remains to show that

$$f(g(x^{t+1})) \leq f^t(x^{t+1}). \tag{23}$$

We will first show that Eq. (23) can be established for specific designs of h making f^t a majoriser of $f \circ g$. More specifically, we will establish the proof for two local models $f_i^t(x) := f(g^t(x)) + \alpha^t h_i(x), i = 1, 2$, with

$$h_1(x, x^t) = \langle x - x^t, \mathbf{H}(x - x^t) \rangle \tag{24}$$

$$h_2(x, x^t) = \langle g^t(x) - g(x^t), \mathbf{H}(g^t(x) - g(x^t)) \rangle \,, \tag{25}$$

where $\mathbf{H} \in \mathbb{R}^{k \times k}$ is some positive definite matrix ensuring the feasibility on $\mathcal{S}$. As a side note, the design choice h_2 is a special case of h_1 with an iteration dependent $\mathbf{H}^t$. To see this, we plug the expression $g^t(x) = g(x^t) + \langle \nabla g(x^t), x - x^t \rangle$ into h_2, yielding $h_2(x, x^t) = \langle \nabla g(x^t)^\top (x - x^t), \mathbf{H} \nabla g(x^t)^\top (x - x^t) \rangle$, from which we can read out that $\mathbf{H}^t = \nabla g(x^t) \mathbf{H} \nabla g(x^t)^\top$. Note that Assumptions i) require $\nabla g(x^t)$ to have full rank, because otherwise $h_2(x, x^t)$ may be zero for $x \neq x^t$.

We now show that for h being either h_1 or h_2, our local model f^t in Eq. (22) is a majoriser of $f \circ g$ and Eq. (23) holds. The proof follows the same argumentation as well-known Majorisation-Minimisation methods [58, Lemma 2.1 & Theorem 2.2]. In essence, from the smoothness of g, we know from [54, Lemma 1.2.3] that there exists a Lipschitz constant $a \in \mathbb{R}$ such that for all $x \in \mathbb{B}^k$ it holds

$$\| g(x) - g(x^t) - \langle \nabla g(x^t), x - x^t \rangle \|_2 \leq a \| x - x^t \|^2. \tag{26}$$

On the other hand, the quadratic function f is Lipschitz continuous on the finite and bounded set $\mathcal{S}$ [62, Theorem 10.4], *i.e.* $\forall x, y \in \mathcal{S}$ we can find $b \in \mathbb{R}$ such that

$$|f(x) - f(y)| \leq b \| x - y \|_2. \tag{27}$$

Substituting $x = g(x)$ and $y = g^t(x)$ in Eq. (27) and using the relation in Eq. (26), we get

$$|f(g(x)) - f(g^t(x))| \leq b \| g(x) - g^t(x) \| \tag{28}$$

$$\leq ab \| x - x^t \|^2, \tag{29}$$

from which follows

$$f(g(x)) \leq f(g^t(x)) + ab \| x - x^t \|^2. \tag{30}$$

Next, we compare the objective in Eqs. (22) and (30). Given that $\mathbf{H}$ is positive definite, it becomes clear that we can find a positive constant α^t and further majorise (30) by f_1^t as

$$f(g(x)) \leq f(g^t(x)) + ab \| x - x^t \|^2 \tag{31}$$

$$\leq f(g^t(x)) + \alpha^t \langle (x - x^t), \mathbf{H}(x - x^t) \rangle = f_1^t(x). \tag{32}$$

This is also true for h_2. Since $\mathbf{H}$ is positive definite and $\nabla g(x^t)$ has full rank, it follows that $\nabla g(x^t) \mathbf{H} \nabla g(x^t)$ is positive definite too. Hence, with a proper choice of α^t, we can further majorise (30) by f_2 as

$$f(g(x)) \leq f(g^t(x)) + ab \| x - x^t \|^2 \tag{33}$$

$$\leq f(g^t(x)) + \alpha^t \langle \nabla g(x^t)^\top (x - x^t), \mathbf{H} \nabla g(x^t)^\top (x - x^t) \rangle = f_2^t(x). \tag{34}$$

Hence, by properly setting the penalty factor α^t we can majorise the true objective. Thus, $f(g(x)) \leq f^t(x)$ for all $x \in \mathbb{B}^k$, and specifically $f(g(x^{t+1})) + \alpha \langle g(x^t), \mathbf{H} g(x^t) \rangle \leq f^t(x^{t+1})$ as desired. Looking back at the argumentation so far we notice that we never had to use any properties of $h_2(x, x^t)$ besides the properties listed in assumption i). The assumption that $h(x^t, x^t) = 0$ is crucial to obtain $f^t(x^t) = f(g(x^t))$. The second condition that $h(x, x^t) > 0$ for all $x \neq x^t$ is important to prove that f^t is a majoriser of $f \circ g$ if α_t is chosen big enough. The general form of the inequalities (32) and (34) is

$$f(g(x)) \leq f(g^t(x)) + ab \| x - x^t \|^2 \tag{35}$$

$$\leq f(g^t(x)) + \alpha^t h(x, x^t) = f^t(x). \tag{36}$$

We can find values for α^t so that this holds, because $h(x, x^t)$ is non-zero for $x \neq x^t$ and since we only consider binary vectors for x and x^t. For arbitrary x with $x \neq x^t$ the function $h(x, x^t)$ could still have approached zero in some limit.

Proof using Assumption ii): Local search argument: Since QuCOOP optimizes over binary vectors x for which $g^t(x) \in \mathcal{S}$, it follows that g and g^t are both valid, possibly different parametrisations of a subset of the feasible set $\mathcal{S}$, and that we have found a better point on $\mathcal{S}$ than $g(x^t)$. If the parametrisation on that point differ so that $g(x^{t+1}) \neq g^t(x^{t+1})$ it is not clear why $g(x^{t+1})$ should also have a better objective value than $g(x^t)$. However if $g(x^{t+1}) = g^t(x^{t+1})$ on all the relevant binary vectors x^t so that $g^t(x^{t+1}) \in \mathcal{S}$ we have the guarantee that $g(x^{t+1})$ has also a better energy than $g(x^t)$.

2. **With the reparametrisation step:**
In the case that the two parametrisations g^t and g differ, as explained in the main text, we can call $\tilde{x}$ the output of the minimisation step in Problem (8) and re-calibrate the iterate as $x^{t+1} = g^{-1}(g^t(\tilde{x}^{t+1}))$. It follows that $f(g(x^{t+1})) \leq f(g^t(x^t)) = f(g(x^t))$ as desired.

□

Note that the re-calibration is also common in classical composite optimisation, *cf*. [58, Composite Gauss–Newton]. In our shape matching and point sets registration problems for the case of permutation matrices, if the iteration step stays inside the set of permutation matrices, then our linearised parametrisation g^t coincides with the original parametrisation g. We will present proof for this result later in this supplement, Lemma 5. This observation saved us the recalibration step and allowed for fast convergence. In the experiments, we also noticed a monotone decrease in the objective function values.

Proof of Lemma 2. We show that any permutation matrix $\mathbf{P}$ of n elements can be written as an ordered product of $k = n(n-1)/2$ binary-parametrised transpositions $\mathbf{P}_i$.

Proof. Let $\mathbf{P}$ be an arbitrary permutation we want to decompose in this way and let c be the cycle notation of $\mathbf{P}$. It is well known that any c can be written as a decomposition of disjoint cycles:

$$c = \prod_j f^{(j)}. \tag{37}$$

We will first prove that an arbitrary cycle can be written as a decomposition in 2-cycles in a fixed order. As the order for the 2-cycles we will w.l.o.g. use

$$((1,2),(1,3),...,(1,n),(2,3),(2,4),....,(n-1,n)). \tag{38}$$

The elements of the ℓ-cycle we want to decompose are denoted as

$$g = (g_1, g_2, ..., g_\ell). \tag{39}$$

We construct the decomposition in an iterative fashion. First, let m be the index of the minimal number in the cycle so that

$$g_m = \min\{g_1, ..., g_\ell\}. \tag{40}$$

The following identity holds for permutations:

$$(g_1, g_2, ..., g_\ell) = (g_m g_{m+1})(g_{m+2}, g_{m+3}, ..., g_\ell, g_1, ..., g_{m-1}, g_{m+1}). \tag{41}$$

One notices that the problem is reduced to finding the decomposition for a $(\ell-1)$-cycle with a minimal element bigger than g_m. Therefore one can repeat the procedure and the 2-cycles will be in the order given by Eq. (38).

The decomposition (39) only uses the elements $g_1, g_2, ..., g_\ell$. We can now find these decompositions for each of the disjoint cycles $f^{(j)}$ that c is decomposed into. Since all the cycles $f^{(j)}$ are disjoint, the 2-cycles a cycle $f^{(i)}$ is decomposed into are disjoint from the 2-cycles a cycle $f^{(j)}$ is decomposed into for $i \neq j$. Since these 2-cycles are disjoint, the product commutes and can be rearranged in the order of Eq. (38). If one has another order of the cycles than the one in Eq. (38), one also has to apply the identity (41) in a way that the final decomposition is in that order.

Finally, the number $k = n(n-1)/2$ of distinct cycles c can be decomposed into is obvious from Eq. (38). □

We have also confirmed via computation that removing a single 2-cycle from the tuple of possible 2-cycles applied in a fixed order results in not reaching the complete $S(d)$ at least for $d \leq 6$.

Proof of Lemma 3. We show that the linear approximate permutation matrices $\mathbf{P}^t(x)$ computed by our algorithm fulfil the row and column sum to 1.

Proof. Each partial derivative of $\mathbf{P}$ can be calculated as

$$\frac{\partial}{\partial x_i}\mathbf{P}(x) = \prod_{j=1}^{i-1}\mathbf{P}_j(x_j)\ \ \frac{\partial}{\partial x_i}\mathbf{P}_i(x_i)\ \prod_{j=i+1}^{k}\mathbf{P}_j(x_j) \tag{42}$$

with $\frac{\partial}{\partial x}\mathbf{P}_i(x_i) = \mathbf{T}_i - \mathbf{I}$. So, it is easy to see that there are exactly two rows and two columns of $\frac{\partial}{\partial x}\mathbf{P}(x)$ that are non-zero and contain exactly one 1 and one -1 each. Thus, for all $x \in \mathbb{B}^k$, it holds

$$\sum_{i=1}^{n}\langle\nabla\mathbf{P}(x^t), x - x^t\rangle_{ij} = 0\ \forall j \tag{43}$$

$$\sum_{j=1}^{n}\langle\nabla\mathbf{P}(x^t), x - x^t\rangle_{ij} = 0\ \forall i. \tag{44}$$

As per definition $\mathbf{P}^t(x) = \mathbf{P}(x^t) + \langle\nabla\mathbf{P}(x^t), x - x^t\rangle$, and since $\mathbf{P}(x^t) \in \Pi_n$ fulfils the row and column sum to 1, the validity of the claim for $\mathbf{P}^t$ is immediate. □

B. QUBO Derivation for QuCOOP

We provide details of the derivation of the QUBO in Eq. (9). For ease of notation, let us first write $g^t(x)$ as

$$g^t(x) = g(x^t) + \left\langle\nabla g(x^t), x - x^t\right\rangle \tag{45}$$

$$= \underbrace{g(x^t) - \nabla g(x^t)^\top x^t}_{=:g} + \underbrace{\nabla g(x^t)^\top x}_{=:g}. \tag{46}$$

Now, we have

$$f^t(x) = f(g^t(x)) \tag{47}$$

$$= \langle g_c + g_x, \mathbf{Q}(g_c + g_x) + \mathbf{c}\rangle \tag{48}$$

$$= (g_c + g_x)^\top\mathbf{Q}(g_c + g_x) + (g_c + g_x)^\top\mathbf{c} \tag{49}$$

$$= g_c^\top\mathbf{Q}g_c + 2g_x^\top\mathbf{Q}g_c + g_x^\top\mathbf{Q}g_x + g_c^\top\mathbf{c} + g_x^\top\mathbf{c} \tag{50}$$

$$= g_x^\top\mathbf{Q}g_x + g_x^\top(\mathbf{c} + 2\mathbf{Q}g_c) + g_c^\top\mathbf{Q}g_c + g_c^\top\mathbf{c}, \tag{51}$$

where the last two terms are independent with respect to the variable x. Now taking the $\arg\min$ and discarding those independent terms, we obtain

$$\arg\min_{x\in\mathcal{X}} f^t(x) = \arg\min_{x\in\mathcal{X}} g_x^\top\mathbf{Q}g_x + g_x^\top(\mathbf{c} + 2\mathbf{Q}g_c), \tag{52}$$

from which we can read out the coupling matrix $\mathbf{Q}^t$ and bias vector $\mathbf{c}$ of the QUBO in Eq. (9).

C. Comparison against Q-Match

An interesting question is how does QuCOOP differs from Q-Match [6] on permutation problems. In Q-Match, one linearizes the inner function by selecting a set of disjoint cycles over which the optimisation is performed. For disjoint cycles, the permutation matrix can be written as a linear function of the binary variables. As formalised is Lemma 4, we observed that with our linearisation $\mathbf{P}^t(x) = \mathbf{P}(x^t) + \langle\nabla\mathbf{P}(x^t), x - x^t\rangle$ and since $\mathbf{P}(x^t) \in \Pi_n$, the minimisers x are precisely those such that $x - x^t$ selects the partial derivatives of disjoint cycles in $\nabla\mathbf{P}(x^t)$, non-disjoint cycles leading to $\mathbf{P}^t(x)$ with higher Frobenius norms that are being penalised. However, unlike Q-Match where the selection of disjoint cycles is done by hand which may be sub-optimal, our algorithm optimally selects the best set of disjoint cycles and optimises over them. We will now give the exact characterisation which permutation matrices can be obtained with our linearisation.

Characterisation of Reachable Valid Permutations

Lemma 4. *The matrix $\mathbf{P}^{t+1}$ obtained from $\mathbf{P}^t$ with QuCOOP is a permutation matrix, if and only if for all indices where x^t differs from x^{t+1} the conjugations of the cycles $T_i^{(-1)}$ of the form*

$$C_i := \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \mathbf{T}_i^{(-1)} \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{53}$$

are disjoint.

Proof. First we observe that if we start at the identity permutation the linearisation only lands on a valid permutation if disjoint cycles are applied. Later the statement will be generalised in the above way for arbitrary x^t. The linearisation can be written in general as

$$\mathbf{P}^{t+1}(x) = \mathbf{P}^t + \sum_i (x_i - x_i^t) \frac{\partial}{\partial x_i} \mathbf{P}(x)|_{x=x} \left(\mathbf{T}_i - \mathbf{I}\right) \prod_{j=i+1}^{k} \mathbf{P}_j(x_j^t) \tag{54}$$

$$= \prod_{j=1}^{k} \mathbf{T}_j^x + \sum_i (x_i - x_i^t) \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i - \mathbf{I}\right) \prod_{j=i+1}^{k} \mathbf{T}_j^x . \tag{55}$$

Now we insert the zero vector for x^t. This yields $\mathbf{P}^{t+1}(x) = \mathbf{I} + \sum_i (x_i) \left(\mathbf{T}_i - \mathbf{I}\right)$. Furthermore, we observe that $\mathbf{T}_i - \mathbf{I}$ has -1 as entries on the diagonal on places where the cycle $\mathbf{T}_i$ acts non-trivially. If two cycles that are chosen are not disjoint then one adds a -2 to a diagonal element of $\mathbf{I}$. Therefore, we can not obtain a permutation matrix in this case.

To generalise this idea we look at the general case described in Eq. (55) and divide by $\mathbf{P}^t$:

$$\left(\prod_{j=1}^{k} \mathbf{T}_j^x\right) \left(\mathbf{P}^t\right)^{-1} = \mathbf{I} + \sum_i (x_i - x_i^t) \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i - \mathbf{I}\right) \left(\prod_{j=i+1}^{k} \mathbf{T}_j^x\right) \left(\prod_{j=k}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right), \tag{56}$$

where the product symbol $\prod_{j=k}^{1}$ indicates that we want to apply the cycles in the reverse order than before. Note that the left side of the equation is only a permutation matrix if $\mathbf{P}^{t+1}$ is a permutation matrix. The right side can be further simplified:

$$\left(\prod_{j=1}^{k} \mathbf{T}_j^x\right) \left(\mathbf{P}^t\right)^{-1} = \mathbf{I} + \sum_i (x_i - x_i^t) \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i - \mathbf{I}\right) \left(\prod_{j=i}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{57}$$

$$= \mathbf{I} + \sum_i (x_i - x_i^t) \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i - \mathbf{I}\right) \left(\mathbf{T}_i^x\right)^{-1} \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{58}$$

$$= \mathbf{I} + \sum_i (x_i - x_i^t) \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i^{1-x} - \left(\mathbf{T}_i^x\right)^{-1}\right) \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{59}$$

$$= \mathbf{I} + \sum_i (x_i - x_i^t)(-1)^x \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i^{(-1)} - \mathbf{I}\right) \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{60}$$

$$= \mathbf{I} + \sum_i (x_i - x_i^t)(-1)^x \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \left(\mathbf{T}_i^{(-1)} - \mathbf{I}\right) \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \tag{61}$$

$$= \mathbf{I} + \sum_i (x_i - x_i^t)(-1)^x \left(\left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \mathbf{T}_i^{(-1)} \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) - \mathbf{I}\right). \tag{62}$$

In Eq. (62), we see that we have the same setting as in the special case where we started with the zero vector. There are positive binary variables $(x_i - x_i^t)(-1)^x \in \{0,1\}$ that tell us if an entry changed from the last binary vector iterate. The new cycle that we consider can be obtained from the old ones through a conjugation:

$$C_i := \left(\prod_{j=1}^{i-1} \mathbf{T}_j^x\right) \mathbf{T}_i^{(-1)} \left(\prod_{j=i-1}^{1} \left(\mathbf{T}_j^x\right)\right). \tag{63}$$

Since conjugation does not change the cycle type, C_i have the same order as the T_i. If C_k, C_l are not disjoint and k, l are indices where x^{t+1} differs from x^t then P^{t+1} cannot be a permutation matrix, because in some element in the diagonal we subtract a -2 from the identity matrix in Eq. (56). □

Proof that the Parametrisation $g(x^t)$ Coincides with the Linearised Parametrisation $g^t(x^t)$

Lemma 5. *Within our setting for permutation matrices if $g^t(x^{t+1})$ is a valid permutation matrix then $g^t(x^{t+1}) = g(x^{t+1})$.*

Proof. The non-linearised parametrisation of the permutation matrices starting from the previous iterate x^t is according to Lemma 2:

$$g^t(x^{t+1}) = \mathbf{P}^{t+1} = \prod_{i=1}^{k} \mathbf{T}_i^x. \tag{64}$$

Using $\mathbf{T}_i^x = \mathbf{I} + x_i(\mathbf{T}_i - \mathbf{I})$ we obtain

$$\mathbf{P}^{t+1} = \prod_{i=1}^{k} \left(\mathbf{I} + x_i^{t+1}\left(\mathbf{T}_i - \mathbf{I}\right)\right) = \prod_{i=1}^{k} \left(\mathbf{I} + x_i^t\left(\mathbf{T}_i - \mathbf{I}\right) + (x_i^{t+1} - x_i^t)\left(\mathbf{T}_i - \mathbf{I}\right)\right). \tag{65}$$

If we factor everything out we obtain:

$$\begin{aligned}\mathbf{P}^{t+1} = \prod_{i=1}^{k} \left(\mathbf{I} + x_i^t\left(\mathbf{T}_i - \mathbf{I}\right)\right) + \sum_{i=1}^{k} (x_i^{t+1} - x_i^t) \prod_{j=1}^{i-1} \left(\mathbf{I} + x_j^t\left(\mathbf{T}_j - \mathbf{I}\right)\right)\left(\mathbf{T}_i - \mathbf{I}\right) \prod_{j=i+1}^{k} \left(\mathbf{I} + x_j^t\left(\mathbf{T}_j - \mathbf{I}\right)\right) \\ + \text{ higher order terms.}\end{aligned} \tag{66}$$

This is exactly the linearised parametrisation plus some higher order terms

$$\mathbf{P}^{t+1} = \mathbf{P}(x^t) + \langle \nabla \mathbf{P}(x^t), x^{t+1} - x^t \rangle + \text{ higher order terms.} \tag{67}$$

Since we are in a setting where $\mathbf{P}^{t+1}$ is a permutation matrix we can make use of Lemma 4. For 2-cycles this states that if m, l are both indices where x^t differs from x^t then

$$\left(\left(\prod_{j=1}^{m-1} \mathbf{T}_j^x\right) \mathbf{T}_m^{(-1)} \left(\prod_{j=m-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) - \mathbf{I}\right)\left(\left(\prod_{j=1}^{l-1} \mathbf{T}_j^x\right) \mathbf{T}_l^{(-1)} \left(\prod_{j=l-1}^{1} \left(\mathbf{T}_j^x\right)\right) - \mathbf{I}\right) = \mathbf{0}. \tag{68}$$

The expression on the left side can be further simplified to

$$\begin{aligned}&\left(\prod_{j=1}^{m-1} \mathbf{T}_j^x\right)\left(\mathbf{T}_m^{(-1)} - \mathbf{I}\right)\left(\prod_{j=m-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right)\left(\prod_{j=1}^{l-1} \mathbf{T}_j^x\right)\left(\mathbf{T}_l^{(-1)} - \mathbf{I}\right)\left(\prod_{j=l-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) \\ &= \left(\prod_{j=1}^{m-1} \mathbf{T}_j^x\right)\left(\mathbf{T}_m^{(-1)} - \mathbf{I}\right)\left(\prod_{j=m}^{l-1} \mathbf{T}_j^x\right)\left(\mathbf{T}_l^{(-1)} - \mathbf{I}\right)\left(\prod_{j=l-1}^{1} \left(\mathbf{T}_j^x\right)^{-1}\right) = \mathbf{0},\end{aligned} \tag{69}$$

assuming $l > m$. Inverting all valid permutation matrices that are multiplied to the expression from the right or left side yields the equivalent equation

$$\left(\mathbf{T}_m^{(-1)} - \mathbf{I}\right)\left(\prod_{j=m}^{l-1} \mathbf{T}_j^x\right)\left(\mathbf{T}_l^{(-1)} - \mathbf{I}\right) = \mathbf{0}. \tag{70}$$

Finally, 2-cycles are their own inverse and

$$(\mathbf{T}_m - \mathbf{I}) \left(\prod_{j=m}^{l-1} \mathbf{T}_j^x \right) (\mathbf{T}_l - \mathbf{I}) = \mathbf{0} \tag{71}$$

will result in terms of higher order in x^{t+1} vanishing.

□

D. Quadratic Assignment

Details on QAPLIB results. We provide further details on the QABLIB experiment from the main text. The number of instances per problem size is provided in Tab. II.

n	12	14	15	16	17	18	20	21	22	24
# Instances	8	2	8	13	2	4	8	1	3	1
n	25	26	27	28	30	32	35	40	50	
# Instances	3	8	1	1	2	3	2	1	1	

Table II. QAPLIB [15] problem sizes n used in our experiments.

For a better inspection of the results in Fig. 2, we have sorted them by problem instances, which we display in Tab. III. It is clear that on most instances, our method achieves the best results among the considered benchmark methods. Noteworthy is that we could find optimal solutions on esc problem instances.

Iterative Local Search. Because our algorithm only approximates the feasible set, some feasible solutions cannot be accessed by solving the sub-problems, which results in the fact that the algorithm may not find the absolute minimiser. We investigate the impact of iterative local search, which consists of applying some random perturbation on the actual iterate to get rid of local minima. First, we consider multiple restarts of the algorithm with a randomly chosen starting point. The returned solution is the one with the lowest energy over the multiple restarts. Second, we add some noise to the iterate x^t by randomly selecting one of its entries and flipping it. This has the effect that the objective function does not monotonically decrease any more but sometimes oscillates. The returned solution is that with the lowest energy over the iterations.

Table IV summarizes our results on 15 selected challenging QAPLIB problem instances [75]. The single restart version with noisy iterates performs better than the compared versions. In practice, increasing the variance of the noise, that is, the number of iterate entries getting flipped, did not improve the results that much, and a variance set too large makes the algorithm diverge.

E. Shape Matching

Failure Cases on FAUST [10]. We present failure cases of the shape matching experiment in Fig. I. Both Q-Match and our method difficulty register shapes with large pose differences. In particular, they cannot fix left and right, front and back vertices flip of the initial linear assignment.

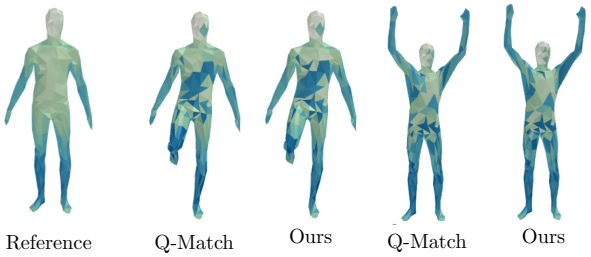


Figure I. Failure shape matching results on the FAUST dataset. The methods partially flip left and right, front and back vertices. This particularly happens on shapes with substantial pose differences.

Experiments on TOSCA Dataset. We perform an experiment on the TOSCA dataset [13]. We register all instances of the cat and dog classes from the dataset.

In the inter-class registration, the source is a cat shape and the targets are dog shapes. Some qualitative results, in line with FAUST results, are presented in Figs. IIa and IIb. On the TOSCA dataset also the methods are non-robust to symmetry flips. We observed failure cases where left and right (*e.g.* left paw registered to right one), front and back (*e.g.* tail registered to head), were flipped, and vice versa.

F. Point Set Registration on D-Wave

We performed point-set registration on D-Wave's quantum annealer (precisely on the Advantage system, see Sec. 7.4 for the specifics) for $n = 3, 5, 7, 10$ points per set. Our results are reported in Fig. III. Up to $n = 5$ and occasionally for $n = 7$, the quantum annealer is able to successfully register the points, and show more difficulties for larger n.

	nug12	nug14	nug15	nug16a	nug16b	nug17	nug18	nug20	nug21	nug22	nug24	nug25	nug27	nug28	nug30
Optimal	578	1014	1150	1610	1240	1732	1930	2570	2438	3596	3488	3744	5234	5166	6124
Ours	612	1048	1196	1676	**1262**	1758	1992	2614	2554	3678	3504	3802	5368	5336	**6222**
Q-Match	608	**1028**	**1182**	**1640**	1264	1816	2014	2650	2574	3682	3608	3966	5486	5342	6408
FAQ	**596**	1054	1186	1660	1282	**1742**	**1946**	**2604**	2580	**3632**	**3500**	**3770**	**5326**	**5284**	6230
2-OPT	620	1040	1216	1704	1288	1870	2004	2738	**2526**	3842	3658	3866	5544	5316	6416

(a) Results on QAPLIB instances in Ref. [57].

	chr12c	chr12b	chr12a	chr15c	chr15a	chr15b	chr18a	chr18b	chr20c	chr20b	chr20a	chr22b	chr22a	chr25a
Optimal	11156	9742	9552	9504	9896	7990	11098	1534	14142	2298	2192	6194	6156	3796
Ours	**12978**	11978	10214	13194	13486	10152	15338	**1534**	**16288**	**2446**	**2456**	**6510**	**6376**	4796
Q-Match	13846	11768	**9916**	**12646**	**12206**	11466	**14466**	1574	25736	3202	3026	6838	6880	**4690**
FAQ	13088	**10468**	33082	16884	19852	**9112**	15440	1712	19836	3206	3166	8582	8920	6744
2-OPT	14636	16748	11370	18634	14234	9404	20960	1710	28800	3616	3918	6810	7114	5502

(b) Results on QAPLIB instances in Ref. [17].

	rou12	rou15	esc16h	esc16i	esc16b	esc16c	esc16d	esc16j	esc16e	esc16a	esc16f	esc16g	rou20	esc32e	esc32g
Optimal	235528	354210	996	14	292	160	16	8	28	68	0	26	725522	2	6
Ours	246244	**368728**	**996**	**14**	**292**	**160**	**16**	**8**	**28**	**68**	**0**	**26**	**740520**	**2**	**6**
Q-Match	**241844**	382094	**996**	**14**	**292**	**160**	**16**	**8**	**28**	**68**	**0**	**26**	762868	**2**	**6**
FAQ	245168	371458	1518	**14**	320	168	62	**8**	30	70	**0**	30	743884	**2**	10
2-OPT	242552	369238	**996**	**14**	**292**	162	**16**	12	30	**68**	**0**	36	785088	**2**	**6**

(c) Results on QAPLIB instances in Ref. [26, 66].

	tai12a	had12	had14	tai15b	tai15a	had16	tai17a	had18	tai20a	had20	tai25a	tai30a	tai35a	tai35b	tai40a
Optimal	224416	1652	2724	51765268	388214	3720	491812	5358	703482	6922	1167256	1818146	2422002	283315445	3139370
Ours	**230704**	1674	2730	**51884360**	402384	3740	512198	5432	730642	7004	1222504	1874474	2514120	296071765	3257058
Q-Match	233040	**1672**	2764	52057859	404700	**3720**	**507218**	5400	742112	**6930**	1222290	1891140	2567762	**287049669**	3291870
FAQ	244672	1674	**2724**	52028170	**397376**	3736	520696	5416	736140	6980	1219484	**1858536**	**2460940**	306237113	**3227612**
2-OPT	246310	1694	2742	51934163	412300	3750	523148	**5394**	**728652**	7016	**1216938**	1888344	2525772	305864564	3340968

(d) Results on QAPLIB instances in Ref. [34, 72, 73].

	scr12	scr15	scr20	bur26f	bur26a	bur26d	bur26h	bur26g	bur26e	bur26b	bur26c	kra32	wil50
Optimal	31410	51140	110030	3782044	5426670	3821225	7098658	10117172	5386879	3817852	5426795	88700	48816
Ours	32696	54926	**111286**	3807270	5446264	**3821372**	7131335	10143927	**5388824**	**3825928**	5430040	**90860**	49272
Q-Match	32758	54684	120824	3815606	5444250	3836955	**7099875**	**10121633**	5399286	3843293	**5427426**	94760	49900
FAQ	40758	**53114**	127150	**3784562**	**5436776**	3822209	7121503	10142604	5398837	3827015	5435069	92930	**49126**
2-OPT	**31884**	57134	118994	3793300	5445951	3823900	7145161	10121687	5433798	3844335	5442586	94360	49194

(e) Results on QAPLIB instances in Ref. [14, 42, 67, 76].

Table III. QAP results on the QAPLIB dataset sorted by instances. The optimal solution and the best solution among the four benchmark methods in rendered in bold. Our method achieves the best solution most frequently on several problem instances: "chr", "esc", "tai" and "bur". In particular, we also achieve optimal solutions on "esc" problem instances. As reported in Ref. [6], Q-Match performs particularly well on "esc" and "had" instances.

G. Annealing and Embedding on D-Wave

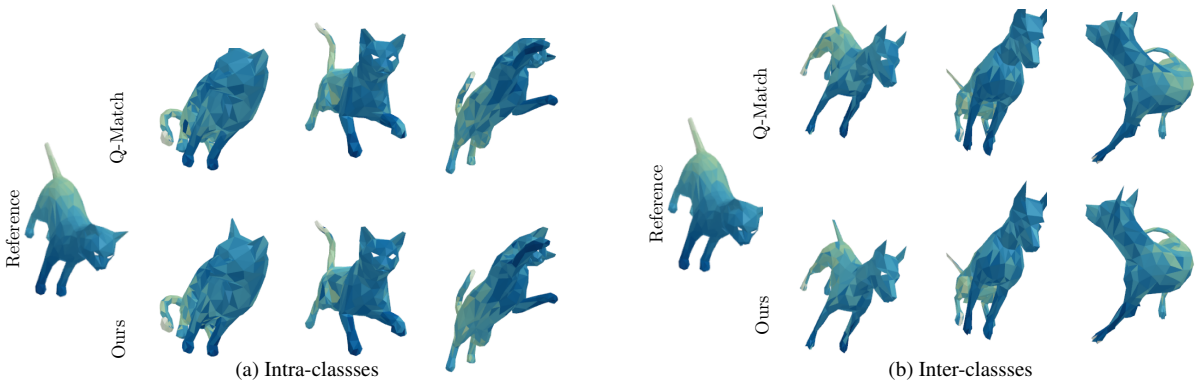
Annealing Process on D-Wave. Programming on D-Wave machines requires defining couplers and biases of the problem and sending them to a quantum processing system. The quantum processor creates a network of logical variables according to the problem size, which is minor-embedded in the quantum hardware. The network starts in a global superposition of all possible basis states. During the quantum annealing, the provided couplers and biases are changed into magnetic fields that deform the state landscape, emphasising the state that is most likely the solution to the underlying optimisation problem.

Minor Embedding. In most cases, our problems result in fully connected logical variables graphs. Embedding of the logical problem onto the quantum hardware often faces the sparse variable connectivity problem. In order to create non-existing physical connections, the QPU *chains* a set of physical variables by setting the strength of their connecting couplers high enough to correlate them.

Fig. IV displays the the embedding on D-Wave of QAP for $n = 5, 10, 15$, as well as the histograms of the sampled energies. Images are obtained from the D-Wave Leap problem inspector. For small n, the histogram looks like a Boltzmann distribution concentrated around the lowest energy. This indicates the confidence of the annealer in the solution

			Number of restarts		
#	Problem	Optimal	50 w/o noise	1 with noise	1 w/o noise
1	chr12c	11156	**11566**	12470	11566
2	rou12	235528	240664	**238954**	245208
3	tai15a	388214	**393476**	400892	394642
4	chr15a	9896	**11052**	11562	12682
5	chr15c	9504	11758	12980	**11212**
6	rou15	354210	367812	**364192**	373132
7	esc16b	292	**292**	**292**	**292**
8	tai17a	491812	**503140**	508222	509066
9	rou20	725522	746180	**734720**	751658
10	chr20b	2298	2688	2764	**2518**
11	tai20a	703482	738124	**733400**	740484
12	chr22b	6194	6828	**6506**	6518
13	tai30a	1818146	1896112	1887748	**1879078**
14	tai35a	2422002	2537102	**2483626**	2524962
15	tai40a	3139370	3277944	**3264446**	3277034

Table IV. Randomness analysis of our algorithm on a 15 selected QAPLIB problems. The two digits in the problem's name stand for the problem size n. Compared are three variants of the algorithm: "50 w/o noise" refers to 50 restarts of the algorithm with different, random starting points and no noise added to the iterates; "1 with noise" refers to a single restart of the algorithm with $x^0 = \mathbf{0}$ and noise added to the iterates; and "1 w/o noise" refers to the standard, single restart of the algorithm with $x^0 = \mathbf{0}$ and no noise added to the iterates. The optimal solution and the best solution among the benchmark variants are rendered in bold. The "50 w/o noise" variant of the algorithm performs only slightly better than the standard run, returning five times the lowest energy compared to four for the standard version. On the other hand, a single restart with noise added to the iterates often performs better than the two other alternatives, returning $8\times$ the lowest energy.



(a) Intra-classses (b) Inter-classses

Figure II. Shape matching registration results on the TOSCA dataset [13].

proposal. For larger n, the histogram is uniformly distributed over several different energy values. This is principally due to long chains, which result in chain breakages. When this happens, the several physical variables in the chain, supposed to represent the same logical variable, get discordant spin configurations, which disturb the overall energy of the system. We tried setting the chain strength higher to avoid chain breakages. While this successfully eliminates chain breakages, we observed that it worsened the overall result of the algorithm.

Fig. V presents, for QAP problem instances, the growth of the number of logical and physically allocated variables as a function of the problem size n. The number of allocated physical variables is about ten times the number of logical variables.

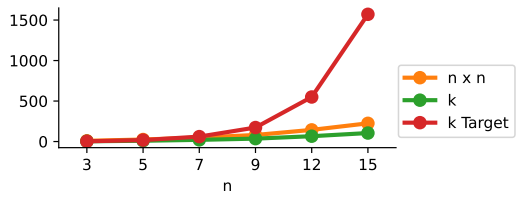
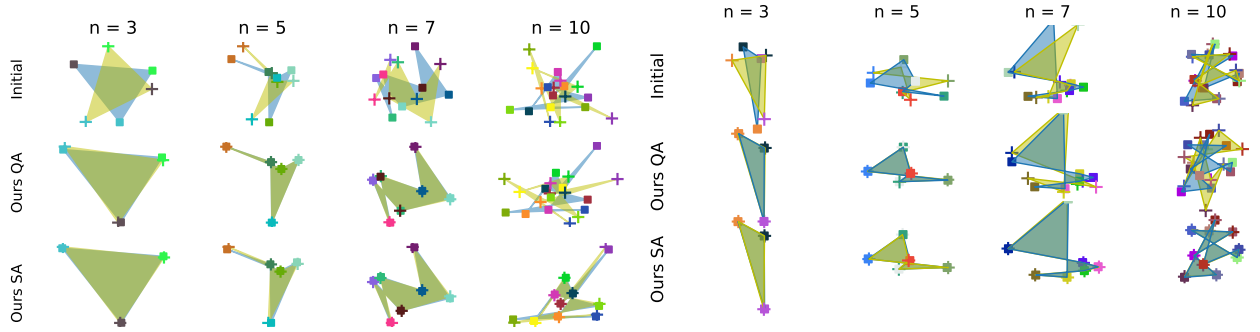


Figure V. Comparing the problem dimension $n \times n$, the number of logical variable $k = n(n-1)/2$ of our problem modelling and the number of corresponding target variables "k Target" allocated on the D-Wave machine for different n on QAP.



(a) 2D (b) 3D

Figure III. Point set registration results computed on D-Wave. Shaded shapes were added for visualisation purposes only. Corresponding points have the same colours, while reference points are marked by squares and template points by crosses. The quantum annealer can register up to $n = 5$ points. From $n = 7$, the quantum annealer can partially recover correct point assignments but fails to perfectly register the points.

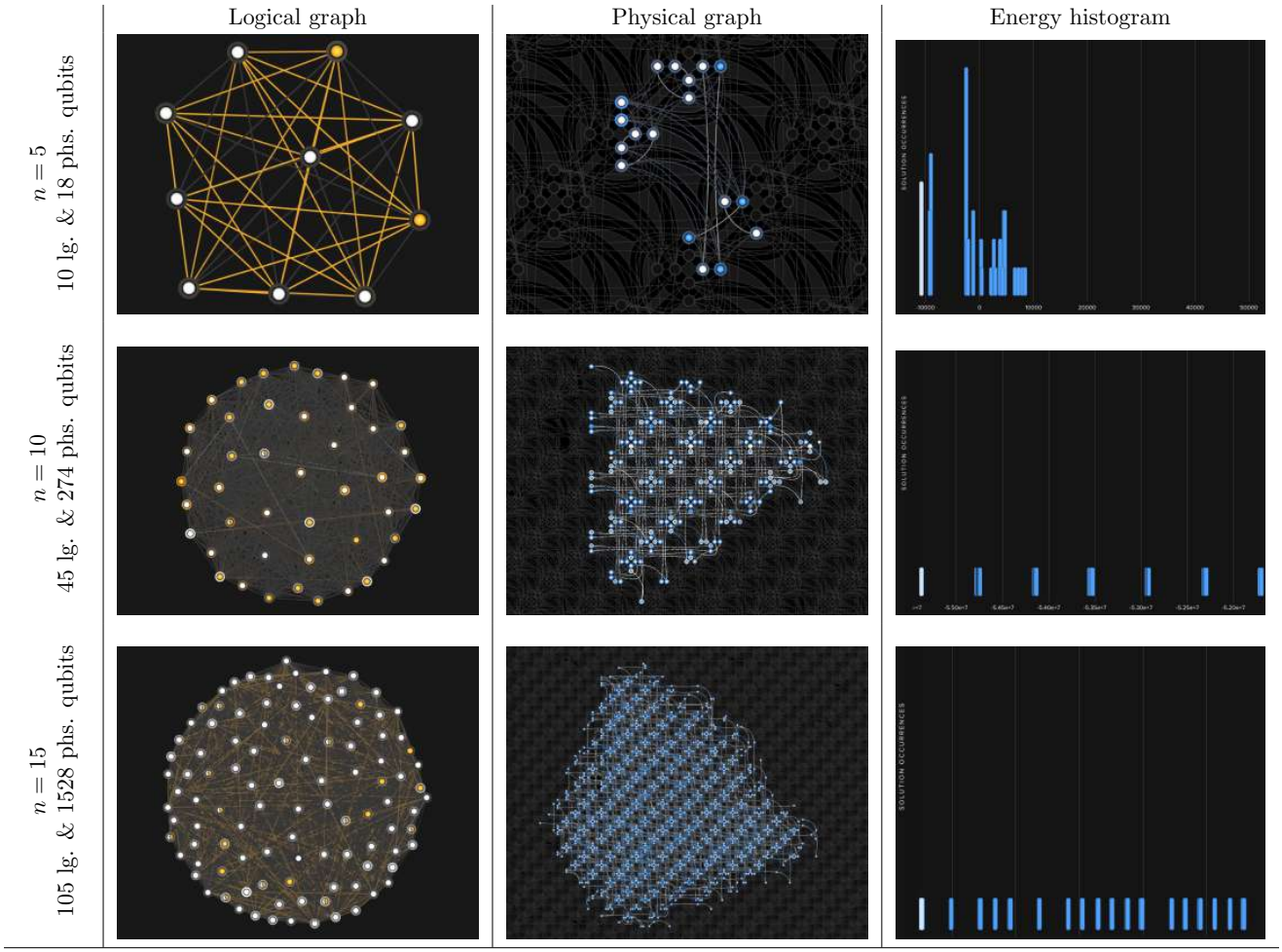


Figure IV. Minor embeddings of QAP on D-Wave for $n = 5, 10$ and 15. The first column shows the logical graph of the problem, the second one visualises the physical embedding of the logical graph, and the third one is the histogram of the returned samples.