Tabu-driven Quantum Neighborhood Samplers

Charles Moussa[1[0000−0002−5387−564X]], Hao Wang[1], Henri Calandra[2], Thomas Bäck[1], and Vedran Dunjko[1[0000−0002−2632−7955]]

[1] LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, Netherlands
[2] TOTAL SA, Courbevoie, France

Abstract. Combinatorial optimization is an important application targeted by quantum computing. However, near-term hardware constraints make quantum algorithms unlikely to be competitive when compared to high-performing classical heuristics on large practical problems. One option to achieve advantages with near-term devices is to use them in combination with classical heuristics. In particular, we propose using quantum methods to sample from classically intractable distributions – which is the most probable approach to attain a true provable quantum separation in the near-term – which are used to solve optimization problems faster. We numerically study this enhancement by an adaptation of Tabu Search using the Quantum Approximate Optimization Algorithm (QAOA) as a neighborhood sampler. We show that QAOA provides a flexible tool for exploration-exploitation in such hybrid settings and can provide evidence that it can help in solving problems faster by saving many tabu iterations and achieving better solutions.

Keywords: Quantum Computing · Combinatorial Optimization · Tabu Search.

1 Introduction

In the Noisy Intermediate-Scale Quantum (NISQ) era [37], hardware is limited in many aspects (e.g., the number of available qubits, decoherence, etc.), which prevent the execution of fault-tolerant implementations of quantum algorithms. Therefore, hybrid quantum-classical algorithms were designed for near-term applications. Examples include algorithms for quantum chemistry problems [31,25], quantum machine learning [7] and combinatorial optimization [18]. They generally consist of one or many so-called parameterized quantum circuits (also known as variational quantum circuits), where the circuit architecture is fixed but the parameters of individual gates are adapted in a classical loop to achieve a computational objective.

Designed for combinatorial optimization, the Quantum Approximate Optimization Algorithm (QAOA) [18] consists of a quantum circuit of a user-specified depth p, involving $2p$ real parameters. To the limit of infinite depth, it converges to the optimum. While numerous works have been studying various theoretical and empirical properties of QAOA [10,13,43,40,32], many practical challenges remain. More practically, only small-sized problems and very limited p can be

run on real hardware, which severly limits the quality of solution obtained empirically [3,5,28]. Last but not least, many open questions still remain, e.g., regarding the comparison of QAOA with other heuristic optimization methods on various cases of instances which stem from particular problem domains, with different optimizers, and with varying levels of experimental (or simulated) noise. One reason of why so many uncertainties remain is that classical simulation is computationally very expensive, and quantum devices are still scarce to prevent large real world tests. [30,3].

In contrast to optimization problems, quantum advantage has been demonstrated in sampling [2]. Indeed, theoretical results establish quantum advantage in producing samples according to certain distributions of constant-depth quantum circuits [11]. In this direction, also in the context of circuits used in QAOA, it has been demonstrated that the sampling of the QAOA circuit, even at $p = 1$, cannot be efficiently simulated classically [19]. The above considerations point to a possibility of utilizing sampling features of QAOA for neighborhood explorations with the added benefit that, since the neighbourhood may be limited to fewer variables, a smaller quantum device may already lead to improved performance of a large instance.

It is interesting to delve into sampling aspects in the domain of classical local search algorithms, where we seek the optimum in the vicinity of the current solution with respect to either the original optimization problem or a subproblem thereof, using a deterministic or stochastic sampling strategy [4]. Such a sampling-based local procedure is typically realized by the combination of some parametric distribution family for drawing local trial points (e.g., the binomial or a power-law distribution [15]) and a selection method for choosing good trial points, and hence the overall outcome of this procedure results in the family of sampling distributions [14,42,27].

In this work, we propose to use QAOA circuits as local neighborhood samplers, having a malleable support in (many) good local optima but still allowing a level of exploration (which is desirable since local optima may not lead to global optima). This introduces the topics of sampling and multiobjective aspects of QAOA that allow balancing between exploration and exploitation. To this end, we study its combination with tabu search (TS), a metaheuristic that has been successfully applied in practice for combinatorial optimization by local search. Moreover, to control the trade-off between exploration and exploitation, we add this critical component in TS to the specification of the standard QAOA circuit.

Contributions - In this work, we construct an algorithm incorporating QAOA in TS with the usual attribute-based short-term memory structure (a.k.a the tabu list). With our approach, we kill two birds with one stone: we gain quantum enhancements, while the local properties of tabu search can make the required quantum computations naturally economic in terms of needed qubit numbers, which is vital in the near-term quantum era. We analyse and benchmark this incorporation with *small QAOA depths* against a classical TS procedure on Quadratic Unconstrained Binary Optimization (QUBO) problems of up to 500 variables. We also propose a penalized version of QAOA incorporat-

ing knowledge from a current solution. We find that QAOA is often beneficial in terms of saved iterations, and can exceptionally find shorter paths towards better solutions. The structure of the paper is as follows. Section 2 provides the necessary background on QUBOs, TS and QAOA. In section 3, we detail the TS procedure incorporating a short-term memory structure with QAOA. The results of our simulations are presented in section 4. We conclude our paper with a discussion in section 5.

2 Background

2.1 QUBO, TS and QAOA

The QUBO formulation can express an exceptional variety of combinatorial optimization (CO) problems such as Quadratic Assignment, Constraint Satisfaction Problems, Graph Coloring, Maximum Cut [26]. It is specified by the optimization problem $min_{x\in\{0,1\}^n} \sum_{i\leq j} x_i Q_{ij} x_j$ where Q is a matrix of real coefficients. This formulation is connected to the task of finding so called "ground states", i.e. configurations of binary labels $\{1, -1\}$ minimising the energy of spin Hamiltonians, commonly tackled in statistical physics and quantum computing, i.e.:

$$\min_{s\in\{-1,1\}^n} \sum_i h_i s_i + \sum_{j>i} J_{ij} s_i s_j \tag{1}$$

where h_i are the biases and J_{ij} the interactions between spins.

The QAOA approach has been designed to tackle CO problems and was inspired from adiabatic quantum computing [18]. As the first step, the classical cost function is encoded in a quantum Hamiltonian defined on N qubits by replacing each variable s_i in eq. 1 by the single-qubit operator σ_i^z:

$$H_C = \sum_i h_i \sigma_i^z + \sum_{j>i} J_{ij} \sigma_i^z \sigma_j^z. \tag{2}$$

H_C corresponds to the target Hamiltonian, and the bitstring corresponding to the ground state of H_C also minimizes the cost function. Second, a so-called *mixer Hamiltonian* $H_B = \sum_{j=1}^N \sigma_j^x$ is leveraged during the procedure. These hamiltonians are then used to build a layer of a quantum circuit with real parameters. This circuit is initialized in the $|+\rangle^{\otimes N}$ state, corresponding to all bitstrings in superposition with equal probability of being measured. Then, applying the layer p times sequentially yields the following quantum state:

$$|\gamma, \beta\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N},$$

defined by $2p$ real parameters $\gamma_i, \beta_i, i = 1...p$ or *QAOA angles* as they correspond to angles of parameterized quantum gates. Such output corresponds to a probability distribution over all possible bitstrings. The classical optimization challenge of QAOA is to identify the sequence of parameters γ and β so as to minimize the expected value of the cost function from the measurement outcome. In the limit of infinite depth, the distribution will converge to the optima.

Tabu Search (TS) [22] is a meta-heuristic that guides a local heuristic search procedure to explore the search space beyond local optimality. One of the main

components of TS is its use of adaptive memory, which creates a more flexible search behavior. Such framework allows using a quantum algorithm as a local search search tool, for solving large instances with limited-sized quantum devices. In the following, we reference a few previous works related to TS for solving QUBOs. This includes a few classical TS strategies and adaptations for incorporating quantum computing. More background on Tabu Search focusing on QUBOs can be found in Appendix.

2.2 Previous work

Various works leveraged TS for solving QUBOs [21,34,23,35] using short-term and long-term strategies used during the search. We note also different hybrid settings that combine a basic TS procedure with another framework such as genetic search [29] and Path Relinking [41]. TS was also incorporated with quantum computers to tackle larger problems beyond their limitations. Indeed, finding methods to leverage smaller devices is of main importance. Many divide and conquer approaches have been designed for quantum circuits and algorithms [17,38,12,36]. In the context of this paper, the size of the QC comes into play more naturally as a hyperparameter defining the ≪radius ≫of the search space.

With respect to the interplay between TS and quantum techniques, to our knowlege TS has only been considered from the perspective of D-Wave quantum annealers. The first approach of this kind is an algorithm called qbsolv [9]. It starts with an initial TS run on the whole QUBO. Then the problem is partitioned into several subproblems solved independently with the annealer. Subproblems are created randomly, by selecting variables. Non-selected ones have their values fixed (clamping values) from the TS solution. The subsolutions are then merged and a new TS is run as an improvement method. The second approach is an iterative solver designed in [39]. At each iteration, a subproblem is submitted to the annealer. The subproblem is obtained by clamping values from a current solution. A tabu list is used in which each element is a list of variables of length k. Each element is kept tabu for a user-defined number of iterations. In contrast in this work, we consider using QAOA in combination with TS.

3 Tabu-driven QAOA sampling

Inspired by the above-mentioned works, we use a simple TS procedure where QAOA is added in the neighborhood generation phase to solve QUBO problems. Note that we could also apply QAOA in more sophisticated frameworks, but a simpler approach is easier for understanding the benefits of QAOA with TS.

Local search algorithms explore a search space by generating sequences of possible solutions which are refined. At each step we generate so-called neighborhood from a current solution. In particular, if we denote the current solution x, a generated neighborhood corresponds to candidates x' that differ by at most k bits. We denote this set as $N_k(x) = \{x' \in \{0,1\}^n | \delta_H(x', x) \leq k\}$, where δ_H denotes the Hamming distance. For a simple one bit-flip generation strategy, this

corresponds to $k = 1$ and TS uses a modified neighborhood due to tabu conditions. Although increasing k could help exploration, the neighborhood generation comes at exponential cost. But this could mean finding better solutions in fewer TS iterations, and thus also in principle overall faster if a fast good method for neighborhood exploration is devised.

This motivates the use of a quantum algorithm as a proxy for exploring $N_k(x)$. Specifically, we will use QAOA which $2p$ real parameters are tweaked in a continuous optimization scheme resulting in a probability distribution on N_k. Increasing p (assuming the optima are found over the parameters) will improve the quality of the output (likelihood of returning an actual global optimum).

However, in the case of local search, a greedy strategy that tends to select the best point in the neighborhood would not only lead to potential stagnation, but also result in longer optimization time (unless the neighborhoods are already the size of the overall problem) [8]. Indeed, one may also consider modifications which impose (various) notions of locality, which are usually not considered in standard QAOA uses where it is used for the entire instance, with the sole goal of finding optima. To this end, we first outline the basic TS procedure generally used to solve QUBO problems [23,29,41]. Then, we show how QAOA can be combined with the latter. Finally, we propose a modification of QAOA that balances between going for the global optimum, and prioritizing local improvements relative to the current TS solution.

3.1 The basic TS algorithm

The basic TS procedure for solving a QUBO with objective function $f(x)$ is described in Alg.1. It uses a simple *tabu list* recording the number of iterations a variable remains tabu during the search. A variable can be set tabu for a fixed number of iterations (denoted Tabu tenure TT) but also with a random tenure (Line 15). Each iteration can be considered as updating a current solution denoted x_{TS}, exploring a modified neighborhood N_1' due to the tabu considerations. Generally, x_{TS} is chosen greedily when evaluating the objective function over candidates $x \in N_1'$.

For large problem instances, there exists an efficient evaluation technique for QUBO solvers leveraging one-bit flip moves [20]. Let $\Delta_x = f(x') - f(x)$ be a move value, that is the effect in objective of going to x' from x. For one-bit flip moves, we denote as $\Delta_x(i)$ the move value upon flipping the i-th variable, which can be computed using only the QUBO coefficients. The procedure records a data structure storing those move values, which is updated after each TS iteration.

Initially, all variables can be flipped (Line 3). At each iteration, the tabu solution x_{TS} is updated by flipping the variable that minimize the objective over the neighborhood obtained by one-bit flip moves over non-tabu variables (Lines 5-7). If the new tabu solution improves over the best recorded solution, the aspiration is activated. In this case, the tabu attribute of the flipped variable is removed. The tabu list is finally updated (Lines 14-17), and iterations continue until the stopping criterion is reached. This can be either a maximum number of TS iterations, and/or a maximum number of TS iterations allowed without

improvement of the best solution (*improvement cutoff*). Having outlined the basic TS procedure, we discuss the integration of QAOA next.

Algorithm 1: The basic TS framework often used for QUBOs over N variables. $Random(rTT)$ samples a random value in $\{1, \cdots, randTT\}$.

Input: An initial solution x_0, Cost function $f(x)$
Parameter : Tabu tenure TT, Random tabu tenure rTT, Stopping criterion
Output: The best solution achieved x^*

$x_{TS} \leftarrow x_0$;
$x^* \leftarrow x_{TS}$;
$Tabu(i) \leftarrow 0, \Delta_x(i) \leftarrow 0$ for $i = 1 \cdots N$;
while *Stopping criterion not reached* **do**
  Compute the one bit-flip move value $\Delta_x(i)$ for $i = 1 \cdots N$ if $Tabu(i) = 0$;
  $j \leftarrow \{k | \Delta_x(k) = \min_{\{i|Tabu(i)=0\}} \Delta_x(i)\}$;
  $x_{TS}(j) \leftarrow 1 - x_{TS}(j)$;
  Update move values $\Delta_x(i)$;
  if $f(x_{TS}) < f(x^*)$ **then**
    $x^* \leftarrow x_{TS}$;
    $aspiration \leftarrow True$;
  else
    $aspiration \leftarrow False$;
  $Tabu(i) \leftarrow Tabu(i) - 1$ for $i = 1 \cdots N$ if $Tabu(i) > 0$;
  if *aspiration = False* **then**
    $Tabu(j) = TT + Random(rTT)$;
  else
    $Tabu(j) = 0$;

3.2 QAOA neighborhood sampling

In the usual TS algorithms, the neighborhood consists of candidates with Hamming distance one relative to the current tabu solution x_{TS}. We note that sometimes considering also neighbors that are at most k-Hamming distance away from x_{TS} helps in finding better solutions. We start from Alg.1 by incorporating and studying QAOA for the neighborhood generation phase. The number k can be set in our case as large as the (limited) number of available qubits in a quantum hardware.

To study the exploration of such neighborhoods, a brute-force generation approach is initially tested, thereafter replaced by QAOA. As stated before, getting the optimum for subproblems in TS may lead to getting stuck during the search. QAOA, by definition, is a flexible framework as an exploration-exploitation tool. On the one hand, QAOA generates better solutions the deeper the circuit (p) , and the better the classical optimization procedure within QAOA is. It is known it can have advantages over various standard algorithms, e.g. Simulated and Quantum Annealing [40]. To extract all advantages from the capacities of QAOA, we can further modulate the distibution over outputs it produces by limited depth or, as we present next, modifications to the QAOA objective to

prioritize a more local behaviour. Such flexibility is important, not only for the exploration-exploitation trade-off as it provides interesting ways fine-tune the algorithm depending on the instance to solve.

First, the choice of variables to run QAOA on needs to be adressed. Considering the $\binom{N}{k}$ possibilities would be intractable. Variables can be chosen randomly but an approach incorporating one-bit flip move values can help in guiding towards an optimum. The k variables can be chosen amongst the non-tabu ones at each step. Plus, this means QAOA is an attempt at improving over the solution one would get with the one-bit flip strategy outlined in Alg.1. One can either select the k variables greedily or add randomness by using the one-flip gains as weights for defining a probability to be chosen. For simplification, we consider the greedy selection based on one-bit flip move values. If we consider the chosen variables that were flipped, the update step of the incremental evaluation strategy can be applied. Let $l \leq k$ be the number of different bits. The newly generated candidate can be considered as a result of l sequential one-bit flips. Thus, l calls to the above-mentioned efficient procedure are required.

A second consideration concerns the tabu strategy for updating the tabu list. We choose to set as tabu the variables amongst the k chosen ones that were flipped. Choosing to flip all chosen ones could be problematic as it could lead to all variables being tabu very early during TS. An aspiration criterion can be used if the new candidate gives the best evaluation found during the search.

Finally, the question of how to run QAOA is of main importance. In our first scenario, QAOA will be run as a proxy for brute-force (with exploration properties) to optimize the subproblem defined over the k chosen variables. This is done by fixing in the QUBO the non chosen ones from the current tabu solution x_{TS}. The depth p of QAOA can be user-defined. In this work, we limit p to 2 to showcase sampling aspects of QAOA at small depth.

Our QAOA-featured TS is outlined in Alg.2. It starts like Alg.1 until Line 9. A set of k variables is chosen (Line 10). QAOA is then run on the new QUBO where we optimize over the chosen variables, by clamping values from x_{TS} on the non-chosen ones. The best sample obtained with QAOA is then extracted (Lines 11-13). If the QAOA solution improves over the current TS solution, the latter is replaced by the former. Otherwise a simple one-bit flip strategy is applied (Lines 14-17). The move values are then updated by l calls of the fast incremental method (Line 18). Finally, the best solution obtained during TS and the tabu list are updated (Lines 19-29).

As previously mentioned, QAOA is generally used in a global optimization strategy. We consider in the next subsection a modification for local search.

3.3 Enforcing locality with penalized QAOA

As a tool in local search algorithms, QAOA may be useful with modifications which impose notions of locality. We incorporate these notions in the cost hamiltonian so that they are captured during the QAOA evolution. This can be done through the cost function by adding a penalty term. A possibility is to consider

Algorithm 2: The adaptation of Alg.1 with QAOA.

Input: An initial solution x_0, Cost function $f(x)$
Parameter : Tabu tenure TT, Random tabu tenure rTT, Size of subproblem k, Stopping criterion
Output: The best solution achieved x^*

$x_{TS} \leftarrow x_0$;
$x^* \leftarrow x_{TS}$;
$Tabu(i) \leftarrow 0, \Delta_x(i) \leftarrow 0$ for $i \in I = \{1, \cdots, N\}$;
while *Stopping criterion not reached* **do**
  $x_{TS,prev} \leftarrow x_{TS}$.;
  Compute the one bit-flip move value $\Delta_x(i)$ for $i \in I$ s.t. $Tabu(i) = 0$;
  $j \leftarrow \{r | \Delta_x(r) = \min_{\{i \in I | Tabu(i)=0\}} \Delta_x(i)\}$;
  $x_{TS,1flip} \leftarrow x_{TS}$;
  $x_{TS,1flip}(j) \leftarrow 1 - x_{TS}(j)$;
  Select greedily or randomly a subset of variables $K \subseteq I$ s.t. $|K| = k$;
  Get a new QUBO by fixing the $N - k$ other variables in $f(x)$ from x_{TS};
  Run QAOA and get the best sample $x_{k,QAOA}$ minimizing the new QUBO;
  $x_{TS,QAOA} \leftarrow \{x^i_{TS}, i \in I \backslash K | x^i_{k,QAOA}, i \in K\}$;
  if $f(x_{TS}) < f(x_{TS,QAOA})$ **then**
    $x_{TS} \leftarrow x_{TS,QAOA}$;
  else
    $x_{TS} \leftarrow x_{TS,1flip}$;
  Update move values $\Delta_x(i), i \in I | x^i_{TS,prev} \neq x^i_{TS}$;
  if $f(x_{TS}) < f(x^*)$ **then**
    $x^* \leftarrow x_{TS}$;
    $aspiration \leftarrow True$;
  else
    $aspiration \leftarrow False$;
  $Tabu(i) \leftarrow Tabu(i) - 1$ for $i \in I$ if $Tabu(i) > 0$;
  for $j \in \{k\} | x^j_{TS,prev} \neq x^j_{TS}$ **do**
    if *aspiration = False* **then**
      $Tabu(j) \leftarrow TT + Random(rTT)$;
    else
      $Tabu(j) \leftarrow 0$;

the Hamming distance with a current tabu solution x_{TS}. Hence, the objective for QAOA becomes:

$$\min_x [f(x) + A\delta_H(x, x_{TS})], \tag{3}$$

where δ_H corresponds to the Hamming distance, and A is a constant. The right-hand side additive term of Eq.3 aims to encourage the output of candidates that differ by few bits from the current solution if $A > 0$, and vice-versa. As a neighborhood sampler, the penalty may help in enforcing locality. However, setting the parameter A is non trivial for activating the effect of the extra term.

There is also a possibility to add information about the fitness gain in differing from x_{TS}. If switching a bit is an improving move from x_{TS}, it would be

prioritized. Our algorithm uses the one-bit flip move values Δ_x in order to select the variables to run QAOA on. These values can be used as coefficients in the penalty term to compose a weighted driving term using the ising values $\{-1, 1\}$ for variables x^j:

$$-\frac{1}{2}\sum_{j=1}^{N}\Delta_x(j)(-1)^{x^j_{TS}}x^j \tag{4}$$

For a minimization problem, $\Delta_x(j) < 0$ characterizes encouraging flipping the j-th variable in new candidates. For a candidate in this case, $\Delta_x(j)$ is added to the cost. Conversely, $\Delta_x(j) > 0$ would result in penalizing candidates with the j-th bit value flipped. One can also multiply by a positive constant A for enforcing more the locality effects. The penalty translates in an additional operator term $H_{penalty}$, following an application of a usual cost operator in QAOA, where:

$$H_{penalty} = -\frac{1}{2}\sum_{j=1}^{N}\Delta_x(j)(-1)^{x^j_{TS}}\sigma_z^j \tag{5}$$

The corresponding quantum circuit of depth one is very simple and given by:

$$\bigotimes_i R_Z((-1)^{x^i_{TS}}\gamma\Delta_x(i)), \gamma \in \mathbb{R}.$$

4 Simulations

We performed extensive simulations over instances of QUBO problems publicly available in OR-LIB [21,6,1]. In designing them, our objectives are 1) investigating whether exploring larger neighborhood can facilitate faster convergence (in terms of TS iterations) under the assumption of finding the optima iteratively, 2) elucidating the effect of locality on QAOA output given by the introduced penalty in Eq. 5, and 3) studying the utility of QAOA as a proxy for brute-force.

4.1 Larger neighborhood exploration benefits

To study the first objective we replaced QAOA by brute-force search in Alg.2. Starting from the all-zero initial solution, we first run Alg. 1 with different constant tabu tenures (randTT = 0). Then, we do the same with brute-force TS for different values of k up to 20. We study two regimes that differ in how TS search results, namely when k is not comparatively small to N on instances where $N = 20$, and when it is on instances of 100, 200, and 500 variables.

***k/N* relatively large** In the first regime, we assume that we can explore a large percentage (more than 25%) of the instance size greedily. As instances, we take the first eight instances of *bqpgka* (named 1a-8a consisting of 30-100 variables), for which we generate solutions using Alg. 1. Then we select randomly 20 variables out of N and clamp values of the non-selected ones from the solutions. We do so five times per instance, resulting in 40 instances of size 20. Then TS

with different k/N values for subproblems $(0.9, 0.75, 0.5, 0.25)$ were tried on this suite. When using brute-force, we set $TT = 2$ and tried many values for Alg.1.

Fig. 1 shows the proportion of (run, target value) pairs aggregated over all functions for 10 targets generated by linear spacing using the benchmarking and profiling tool IOHprofiler [16] for iterative optimization heuristics. The target values were normalized by the optimum of the problems. We observed that for $k/N \geq 0.5$, these instances are straightforward to solve (in 3 iterations). The case $k = 5$ required 8 iterations on one instance but managed to achieve optimality on all of them. However Alg.1, run for 20000 iterations, failed to solve the same instance. Letting this instance aside, 5 iterations would be required for $k = 5$, and 22 for the classical TS procedure. Hence, we clearly observe, as expected, degrading performances as k/N decreases. This also enabled us to confirm numerically that a flip-gain based approach when considering subproblems is in general beneficial towards solving QUBOs.

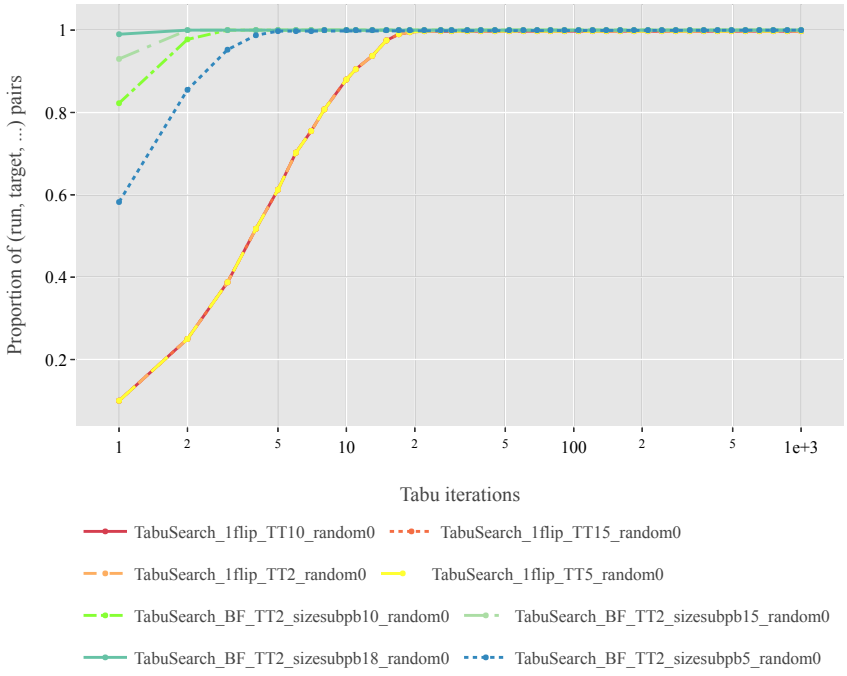

Fig. 1. Aggregated over all 20-variable problems, the empirical cumulative distribution function (ECDF) of tabu iterations is shown for each algorithm. To compute the ECDF, we take 10 target values that are evenly spaced between the smallest and largest function values observed in each pair of dimension and problem. Lines corresponding to runs of Alg. 1 (labeled by "TabuSearch_1flip_") superpose for different values of TT. We see that increasing k facilitate faster convergence.

***k/N* relatively small** In this regime, we study the case when k/N is relatively small (less than 20%). The simulations are carried out on 15 QUBO instances from bqpgka: 100-variable (named 1d-5d), 200-variable (1e-5e) and 500-variable problems (1f-5f). Algorithms are run until the best objective value found in [21] is reached or a maximum number of TS iterations is reached.

Tables 2, 3 and 6 display our results for $k = 10, 15, 20$ and for $TT = 5, 10, 15$. For Alg.1, we limit the number of iterations to 20000, 1000 for the brute-force approach (200 for $k = 20$ though). In these cases, we observe that we can find a k such that better solutions are found using less iterations. In general, TT should not be set large when exploring larger neighborhood. For instance, the case $TT = 15$ started to favor the basic TS in terms of target achieved.

We also run the algorithms with different TT values ranging from 2 to 10 and adding 15. Tables 1,4 and 5 show for the different values of TT, which ones achieved the best performances in terms of target only. We note promising performances, especially on the most dense instances 4f and 5f. These instances, when considering the underlying graph given by the coefficients connecting different variables, have a density of respectively 0.75 and 1 (a non-zero coefficient for each pair of variables). For $k = 20$ the proposed approach achieved optimality where Alg.1 failed. Again, we observe performances, in terms of target achieved, depend on setting well the tabu tenure in accordance with k. Intuitively, one could think that the larger k, the smaller TT has to be to save iterations and achieve a better objective value. But we clearly observe counter-examples.

Larger k exploration, in this regime, turns out to not always be beneficial. This seems conterintuitive when considering a target objective only, on an instance to instance basis comparison. Fig. 2 shows the proportion of (run, target value) pairs aggregated over all functions for 1000 targets generated by linear spacing. The target values were normalized by the optimum of the problems. Again, we observe that, with larger k, the proportion of successes is higher, when measured at the same number of iterations. Note that this can only be observed for $k = 20$ up to 200 iterations. The proportion for the basic TS was close to 0.9 while the brute-force approach was superior to 0.99. Hence, we can reach very good solutions with less iterations as k increases.

In summary, as opposed to the previous regime, the structure of the problems becomes very important that we have to look at performances in an aggregated way to witness the benefits of exploring larger neighborhood. Having outlined some performances given by the brute-force approach on subproblems, we switch to QAOA, and study its sampling effect as a proxy.

4.2 QAOA as a proxy for brute-force

The second part of our simulations studies the output of QAOA as a proxy for brute-force. To this end, we first study an example TS run from our previous simulations. We take the subproblem QUBOs obtained at each step (except the first one), and run QAOA at $p = 1$ and 2, and we study the distribution of the energy given by $|\gamma, \beta\rangle$, after optimization, with and without the penalty term.

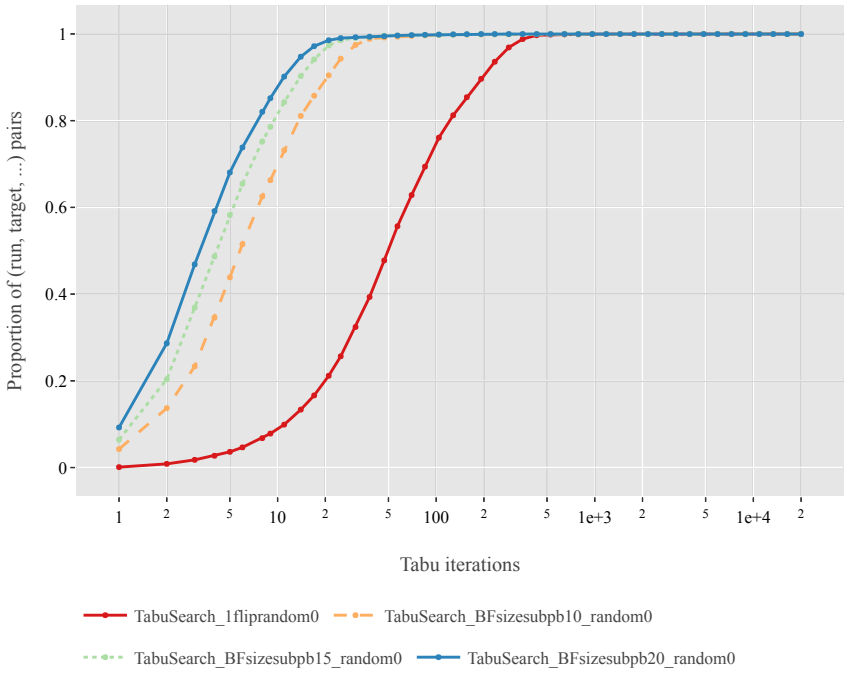

Fig. 2. Aggregated over all dimensions (100, 200, and 500) and all problems, the empirical cumulative distribution function (ECDF) of tabu iterations is shown for each algorithm. To compute the ECDF, we take 1000 target values that are evenly spaced between the smallest and largest function values observed in each pair of dimension and problem.

Having outlined the properties of the QAOA output, we run Alg. 2, and study its performances in comparison to Alg. 1. From the optimized angles, we try different sampling strategies to generate a candidate per iteration: just sampling once, sampling 10 times and choosing a candidate greedily, and finally consider all samples (even during optimization) greedily. The latter corresponds to a quasi brute-force (BF) approach.

Energy distribution of QAOA As a first step, we study how the QAOA output distribution looks like at small depth, with the purpose of elucidating how it can help avoiding detrimental greedy search behavior. We consider, as an example, instance 1e for which $k = 15, TT = 5$ used 13 iterations greedily. The subproblem QUBOs are kept and we run QAOA on them as previously stated. The third and last iteration are interesting when considering the penalty term. The former is a case where the optimum is obtained by flipping all bits except one and where all flip moves are favorable. The last iteration has flip moves from the current tabu solution discourage flipping all bits. Plus, very few

candidates (0.2%) improve over the tabu solution. For these iterations, we look at the quantum state given by QAOA and analyse the distribution of the QUBO evaluation (or energy in an Ising context).

Fig. 4 shows the distribution given by 10^5 samples from the last iteration's quantum state at $p = 1$ and 2. At $p = 1$, we observe a homogeneous spread with two major humps on the left and right side of the tabu solution evaluation. There is a probability of 23.9% of improving from it by the quantum state. At $p = 2$, we see the distribution being shifted to lower energies, yielding an improvement probability of 75%. The average energy is 16350 for QAOA $p = 1$ and 16428 for $p = 2$. Moreover, the standard deviation of the output decreases from 79.2 to 42. This is expected as to the limit of infinite depth, QAOA converges to the optimum with less variance.

Fig. 4 shows also the distribution of the QUBO evaluation after running simulated annealing (SA) 1000 times with a temperature of 17.5 using 100 steps. We observe that unlike QAOA, the energy spread is restrained to a few points, the optimum being most present. Decreasing slightly the temperature or the number of steps would always yield the optimum. However, in terms of exploration opportunities, QAOA could allow visiting different paths that may lead to fewer iterations required towards improved solutions.

Theory shows that increasing the depth would permit QAOA to find the optimum assuming optimal parameters are found. But by limiting the depth, we can control how other good candidates are spread from the optimum. This could engender new paths to solve a problem differently, where a suboptimal solution on a subproblem leads to an easier one for QAOA towards better candidates. Note this can be done in different ways. One way would be using brute-force and perturbating or mixing the solution, which is not efficient. We could also have the same effect with simulated annealing. But in many cases, we can fail finding the optimum, and even less finding a bunch of candidates around it. Finally, due to its flexibility, QAOA permits to leverage modifications to introduce locality notions in a multiobjective scheme for local search, such as the previously mentioned penalty. In the following, we study its effect on the QAOA distribution.

Penalty effect In section 3.3, we introduced a penalty term to impose notions of locality in QAOA as a local search tool. An extra operator based on the hamiltonian given in eq.5, translates to a circuit of depth one concatenated with a QAOA layer. We study its effect on the QAOA distribution obtained on the resulting sub-qubos at the third and last iteration.

For iteration 3, Fig. 5 shows that for both QAOA and its penalized version, the distribution tends to output candidates with largest Hamming distances with x_{TS} as expected. Also, we see that the most likely candidate is the one that completely differs from x_{TS}, which is more favored by the penalty effect. No significant changes were observed when penalizing at $p = 2$.

For the last iteration, Fig. 6 shows a different type of distributions. The original QAOA at $p = 1$ results in many probability peaks compared to the penalized version, which evolve to a major peak with an increased depth. The

penalized version demonstrates two major peaks, from which the optimum and a close candidate to x_{TS} are preferred.

This characterizes the interplay between optimizing and penalizing. At $p = 1$, the penalized version has a better probability of improving from x_{TS} (0.34 vs 0.239) and a higher probability of finding the optimum (0.1 vs 0.02). However, the unpenalized $p = 2$ version was more likely to output the optimum (respectively 0.75 and 0.26, where the penalty at $p = 2$ yields 0.62 and 0.26).

In summary, using the penalty creates a balance between the greedy approach and one-bit flip gains knowledge from the current solution. This could result in smoothening the distribution while favouring interesting candidates for both objectives. This will modify the search path taken during TS depending on the outcome. Having studied numerically the output of the quantum state one can get with QAOA, and the penalty effect, we switch to less idealized simulations where the subproblems depend on the QAOA output during the TS search.

QAOA exploration possibilities After looking at examples of QAOA output and outlining a few possible exploration opportunities, we carried out a few extra simulations but not considering the subproblems obtained with brute-force. Hence QAOA (and its penalized version) was called once per iteration with BIPOP-CMAES optimizing from one set of angles. In the following, we give a few examples to illustrate the exploration possibilities.

We take instances where BF simulations required few iterations and where the basic TS was beaten in target. Namely, instance 1d for $k = 15$ and $TT = 5$ and 1e for $k = 15$ and $TT = 10$. We run Alg. 2 10 times, limiting them to 20 TS iterations. Different numbers of samples are used for generating a new candidate per iteration: just once, 10 times and 1000 times. The latter could be considered as a quasi-brute-force approach in these runs. We denote as m the number of samples used in the following.

Fig. 3 and Fig. 7 show the median of the best normalized evaluation obtained per run by iteration. We observe from iteration 7 for 1d and 10 for 1e that $m = 1000$ is equivalent (in median) to the BF generation. In general, the more samples used, the better the solution found. However, we had at a few iterations median runs that achieved higher values than BF. For instance, this happened for the penalized $p = 2$ QAOA at the 5th iteration with $m = 10$ for 1d and $m = 1000$ at the 12th for 1e.

We consider also the frequency of runs for which the basic TS was beaten, and the optimum was found. Tables 7 and 8 summarize our results. Increasing m improves the frequency of successful runs. We observe that new paths were found, mainly with an extra iteration or two but exceptionally one run or two over 10 could save one iteration. This was the case on instance 1e where 12 iterations were used instead of 13, exclusively with the penalized version.

These examples are numerically in favor of a greedy (or quasi) approach in solving subproblems. However, QAOA allows, through a trade-off between exploration and exploitation, discovering new paths towards optimality that are still interesting in terms of number of iterations.

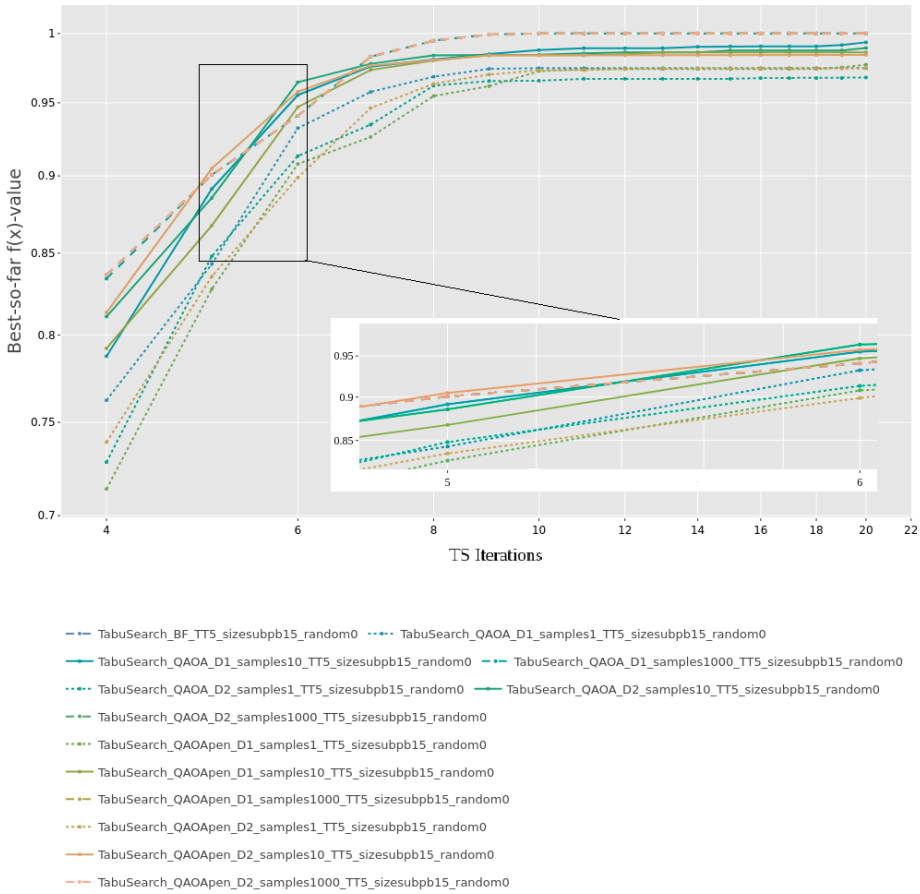

Fig. 3. Median of best normalized evaluations achieved over TS iteration for instance 1d. The ≪BF ≫mention means brute-force generation while D1, D2, mean respectively $p = 1, 2$ and the penalized version is indicated by ≪pen ≫. A higher curve corresponds better solutions reached by TS iteration. At iteration 20, the basic TS value would be 0.38, while the lowest QAOA curve value is 0.967. The 1000 samples runs and BF are over the 0.99 starting at the 8th iteration. At iteration 5, the penalized $p = 2, 10$ measurements version is slightly better than the others, even BF (0.9049 vs 0.9007). At the 6th, the $m = 10$ versions are above BF (respectively 0.9645 and 0.9578 for unpenalized and penalized $p = 2$, 0.9555 and 0.9470 for $p = 1$, and 0.9409 for BF).

5 Conclusion and outlook

In this work, we studied sampling aspects when quantum approaches, specifically the QAOA algorithm, are considered in combinatorial optimization. We considered a practically relevant setting where a gate-based quantum algorithm, limited in the number of qubits, is utilized in a hybrid quantum-classical framework to solve large optimization instances faster. Our framework constitutes a powerful yet simple heuristic, Tabu Search, in tandem with QAOA as a local neighborhood sampler.

As a starting point, numerical experiments over open-source QUBO problems up to 500 variables validate using QAOA as a proxy to explore larger neighborhood, under the assumption that subproblems are solved optimally. Continuing, we investigated the exploration possibilities given by QAOA output at small depth. User-defined parameters such as depth and number of measurements used to generate a candidate, can be increased to favor exploitation. On our examples, solving subproblems emphasizing more on the latter gave better general performances. Yet, we found that exploration can be beneficial. Iterations can be saved with our QAOA procedure, illustrating that missing to generate the solution of a subproblem in previous iterations could yield to faster paths towards better solutions. Hence, the QAOA-based algorithm we introduce in this work becomes a very flexible tool in such hybrid quantum-classical settings.

We see numerous possibilities for future work. First, our model allows for many hyperparameters whose function needs to be explored, and, as is usually done in many local search methods, the exploration/exploitation trade-offs can be made online-adaptive. Second, the effect of real world limitations, most importantly noise,and hardware connectivity, calls for further investigation. We note that for the cases of limited connectivity [3], our algorithm could be adaptive in choosing whether to call a quantum subroutine [32]. Finally, it would be interesting to propose different frameworks (e.g. [34,35,23,29]) with special emphasis on the exploration possibilities given by small-depth quantum algorithms. We believe our approach combined with these types of analyses will provide new promising ways to maximize the use of limited near-term quantum computing architectures for real world and industrial optimization problems.

Acknowledgements

CM, TB and VD acknowledge support from Total. This work was supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037). This research is also supported by the project NEASQC funded from the European Union's Horizon 2020 research and innovation programme (grant agreement No 951821).

References

1. QUBO instances link - file bqpgka.txt, howpublished = http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html

2. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G.S.L., Buell, D.A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M.P., Hartmann, M.J., Ho, A., Hoffmann, M., Huang, T., Humble, T.S., Isakov, S.V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P.V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J.R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M.Y., Ostby, E., Petukhov, A., Platt, J.C., Quintana, C., Rieffel, E.G., Roushan, P., Rubin, N.C., Sank, D., Satzinger, K.J., Smelyanskiy, V., Sung, K.J., Trevithick, M.D., Vainsencher, A., Villalonga, B., White, T., Yao, Z.J., Yeh, P., Zalcman, A., Neven, H., Martinis, J.M.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5, https://doi.org/10.1038/s41586-019-1666-5
3. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Boixo, S., Broughton, M., Buckley, B.B., Buell, D.A., Burkett, B., Bushnell, N., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Demura, S., Dunsworth, A., Farhi, E., Fowler, A., Foxen, B., Gidney, C., Giustina, M., Graff, R., Habegger, S., Harrigan, M.P., Ho, A., Hong, S., Huang, T., Ioffe, L.B., Isakov, S.V., Jeffrey, E., Jiang, Z., Jones, C., Kafri, D., Kechedzhi, K., Kelly, J., Kim, S., Klimov, P.V., Korotkov, A.N., Kostritsa, F., Landhuis, D., Laptev, P., Lindmark, M., Leib, M., Lucero, E., Martin, O., Martinis, J.M., McClean, J.R., McEwen, M., Megrant, A., Mi, X., Mohseni, M., Mruczkiewicz, W., Mutus, J., Naaman, O., Neeley, M., Neill, C., Neukart, F., Neven, H., Niu, M.Y., O'Brien, T.E., O'Gorman, B., Ostby, E., Petukhov, A., Putterman, H., Quintana, C., Roushan, P., Rubin, N.C., Sank, D., Satzinger, K.J., Skolik, A., Smelyanskiy, V., Strain, D., Streif, M., Sung, K.J., Szalay, M., Vainsencher, A., White, T., Yao, Z.J., Yeh, P., Zalcman, A., Zhou, L.: Quantum approximate optimization of non-planar graph problems on a planar superconducting processor (2020)
4. Bäck, T.: Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press (1996)
5. Barkoutsos, P.K., Nannicini, G., Robert, A., Tavernelli, I., Woerner, S.: Improving variational quantum optimization using cvar (2019)
6. Beasley, J.E.: Or-library: Distributing test problems by electronic mail. The Journal of the Operational Research Society **41**(11), 1069–1072 (1990), http://www.jstor.org/stable/2582903
7. Benedetti, M., Lloyd, E., Sack, S., Fiorentini, M.: Parameterized quantum circuits as machine learning models (2019). https://doi.org/10.1088/2058-9565/ab4eb5, arXiv:1906.07682
8. Beyer, H.: The theory of evolution strategies. Natural computing series, Springer (2001)
9. Booth, M., Reinhardt, S.P.: Partitioning optimization problems for hybrid classical / quantum execution technical report (2017)
10. Brandão, F.G.S.L., Broughton, M., Farhi, E., Gutmann, S., Neven, H.: For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances (2018), arXiv:1812.04170
11. Bravyi, S., Gosset, D., König, R.: Quantum advantage with shallow circuits. Science **362**(6412), 308–311 (2018). https://doi.org/10.1126/science.aar3106, https://science.sciencemag.org/content/362/6412/308

12. Bravyi, S., Smith, G., Smolin, J.: Trading classical and quantum computational resources (2015). https://doi.org/10.1103/PhysRevX.6.021043
13. Crooks, G.E.: Performance of the quantum approximate optimization algorithm on the maximum cut problem (2018), https://arxiv.org/abs/1811.08419
14. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the (1+(λ, λ)) genetic algorithm. Algorithmica **80**(5), 1658–1709 (2018). https://doi.org/10.1007/s00453-017-0354-9, https://doi.org/10.1007/s00453-017-0354-9
15. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Bosman, P.A.N. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017. pp. 777–784. ACM (2017). https://doi.org/10.1145/3071178.3071301, https://doi.org/10.1145/3071178.3071301
16. Doerr, C., Wang, H., Ye, F., van Rijn, S., Bäck, T.: Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. arXiv e-prints:1810.05281 (Oct 2018), https://arxiv.org/abs/1810.05281
17. Dunjko, V., Ge, Y., Cirac, J.I.: Computational speedups using small quantum devices (2018). https://doi.org/10.1103/PhysRevLett.121.250501
18. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm (2014)
19. Farhi, E., Harrow, A.W.: Quantum supremacy through the quantum approximate optimization algorithm (2016)
20. Glover, F., Hao, J.K.: Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems. Int. J. Metaheuristics **1** (01 2010). https://doi.org/10.1504/IJMHEUR.2010.033120
21. Glover, F., Kochenberger, G., Alidaee, B.: Adaptive memory tabu search for binary quadratic programs. Management Science **44**, 336–345 (03 1998). https://doi.org/10.1287/mnsc.44.3.336
22. Glover, F.W.: Tabu Search, pp. 1537–1544. Springer US, Boston, MA (2013). https://doi.org/10.1007/978-1-4419-1153-7˙1034, https://doi.org/10.1007/978-1-4419-1153-7_1034
23. Glover, F.W., Lü, Z., Hao, J.K.: Diversification-driven tabu search for unconstrained binary quadratic problems. 4OR **8**, 239–253 (2010)
24. Hansen, N.: Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In: ACM-GECCO Genetic and Evolutionary Computation Conference. Montreal, Canada (Jul 2009), https://hal.inria.fr/inria-00382093
25. Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J., Gambetta, J.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature **549**, 242–246 (09 2017). https://doi.org/10.1038/nature23879
26. Kochenberger, G.A., Glover, F.: A Unified Framework for Modeling and Solving Combinatorial Optimization Problems: A Tutorial, pp. 101–124. Springer US, Boston, MA (2006). https://doi.org/10.1007/0-387-29550-X˙4, https://doi.org/10.1007/0-387-29550-X_4
27. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input–output sequences. Soft Computing **15**(9), 1675–1687 (2011)
28. Li, L., Fan, M., Coram, M., Riley, P., Leichenauer, S.: Quantum optimization with a novel gibbs objective function and ansatz architecture search (2019)
29. Lü, Z., Glover, F.W., Hao, J.K.: A hybrid metaheuristic approach to solving the ubqp problem. Eur. J. Oper. Res. **207**, 1254–1262 (2010)

30. Medvidovic, M., Carleo, G.: Classical variational simulation of the quantum approximate optimization algorithm (2020)
31. Moll, N., Barkoutsos, P., Bishop, L.S., Chow, J.M., Cross, A., Egger, D.J., Filipp, S., Fuhrer, A., Gambetta, J.M., Ganzhorn, M., Kandala, A., Mezzacapo, A., Müller, P., Riess, W., Salis, G., Smolin, J., Tavernelli, I., Temme, K.: Quantum optimization using variational algorithms on near-term quantum devices (2017). https://doi.org/10.1088/2058-9565/aab822, arXiv:1710.01022
32. Moussa, C., Calandra, H., Dunjko, V.: To quantum or not to quantum: towards algorithm selection in near-term quantum optimization (2020). https://doi.org/10.1088/2058-9565/abb8e5
33. niko, Akimoto, Y., yoshihikoueno, Brockhoff, D., Chan, M., ARF1: Cma-es/pycma: r3.0.3 (Apr 2020). https://doi.org/10.5281/zenodo.3764210, https://doi.org/10.5281/zenodo.3764210
34. Palubeckis, G.: Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Annals of Operations Research **131**, 259–282 (10 2004). https://doi.org/10.1023/B:ANOR.0000039522.58036.68
35. Palubeckis, G.: Iterated tabu search for the unconstrained binary quadratic optimization problem. Informatica (Vilnius) **2** (01 2006)
36. Peng, T., Harrow, A., Ozols, M., Wu, X.: Simulating large quantum circuits on a small quantum computer (2019). https://doi.org/10.1103/PhysRevLett.125.150504
37. Preskill, J.: Quantum Computing in the NISQ era and beyond. Quantum **2**, 79 (Aug 2018). https://doi.org/10.22331/q-2018-08-06-79, https://doi.org/10.22331/q-2018-08-06-79
38. Rennela, M., Laarman, A., Dunjko, V.: Hybrid divide-and-conquer approach for tree search algorithms (2020)
39. Rosenberg, G., Vazifeh, M., Woods, B., Haber, E.: Building an iterative heuristic solver for a quantum annealer. Computational Optimization and Applications **65**, 845–869 (2016)
40. Streif, M., Leib, M.: Comparison of qaoa with quantum and simulated annealing (2019), arXiv:1901.01903
41. Wang, Y., Lü, Z., Glover, F.W., Hao, J.K.: Path relinking for unconstrained binary quadratic programming. Eur. J. Oper. Res. **223**, 595–604 (2012)
42. Watson, R.A., Jansen, T.: A building-block royal road where crossover is provably essential. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'07). pp. 1452–1459. ACM (2007). https://doi.org/10.1145/1276958.1277224, https://doi.org/10.1145/1276958.1277224
43. Zhou, L., Wang, S.T., Choi, S., Pichler, H., Lukin, M.D.: Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices (2018), arXiv:1812.01041

Appendix

Tabu Search

Tabu Search (TS) [22] is a meta-heuristic that guides a local heuristic search procedure to explore the search space beyond local optimality. One of the main components of TS is its use of adaptive memory, which creates a more flexible

search behavior. TS begins the same way as ordinary local or neighborhood search, proceeding iteratively from one solution to another until a termination criterion is reached. Each solution x has an associated neighborhood $N(x)$ and each new candidate $x' \in N(x)$ is reached from x by an operation called *move*. x' is said to be a neighbor of x. For unconstrained binary problems, the common move is the simple one-bit flip from which x' differs from x by flipping one variable. TS can be contrasted with a simple descent method where the goal is to minimize the cost function. Such a method only permits moves to neighbor solutions that improve the current objective function value and ends when no improving solutions can be found. The evident shortcoming of a descent method is that this causes the search to stagnate in a local optimum.

Moves that deteriorate the current objective function value are accepted in order to escape local optima. The hope is to explore unvisited neighborhood and potentially reach better optima. The core guidance mechanism in TS is due to the use of memory structures that yield subsequent solutions belonging to a modified neighborhood $N^*(x)$. In all TS implementations, a short-term memory (STM) is used to avoid revisiting recently obtained solutions or using moves that yield them. For instance, one can set as tabu the index of the variable that was flipped to get a current solution. This variable will not be flipped again in the next neighborhood generation. In other words, the modified neighborhood is the result of maintaining a selective history of the moves used during the search. A user-defined parameter *tabu tenure* specifies the number of TS iterations the variable is kept tabu. Such tabu rule can be removed, for instance if flipping a variable led to the best solution found during the search. This mechanism is designed as an aspiration criterion. An example of TS algorithm often used in QUBOs with one-bit flip moves is presented in Alg. 1.

In general, STMs are sufficient to produce high-quality solutions. However, TS can be improved by including long-term considerations. Long-term memory (LTM) is often implemented using frequency-based information, complementing recency-based memory like STMs for selecting moves. For instance, one can record transition counts (e.g., number of iterations a variable has been flipped) or residence counts (e.g., number of iterations a variable was kept unchanged in best generated neighbors). We can use a LTM to reset the search and generate a new initial solution to start from scratch a new search, using accumulated knowledge during the previous run.

In conclusion, a key element of the adaptive memory framework of TS is to create a balance between search exploitation and exploration (also called intensification and diversification). Exploitation strategies encourage move combinations and solution features historically found good. Exploration encourages the search of solutions that differ from those previously visited.

QAOA optimization procedure

To optimize the QAOA angles, we use BIPOP-CMAES [33,24] and run circuits with 1000 measurements to estimate expectation values. The optimizer stops when it has reached 2000 evaluations. BIPOP-CMAES obtained great performances in terms of averaged ratios (as the evaluations divided by the optimum of the subproblem), superior to 0.97 at the considered depths.

Results of experiments

Table 1. Best values of tabu tenure (ranging from 2 to 10, adding 15) achieving the optimum obtained with the first TS iteration to reach the corresponding maximum given in [21] on instances 1d to 5d. The best performances per instance are highlighted in bold. The mention All means all TT values reached the same solution, an inequality refers to all TT values satisfying it. $2 + 5$ means $TT = 2$ and $TT = 5$.

Algorithm	1d (6333)	2d (6579)	3d (9261)	4d (10727)	5d (11626)
Basic TS	15 300	All 71	<9 90	2 67	6 171
TS $k = 10$	All 13	>2 61	6 39	2 31	All 19
TS $k = 15$	All 10	**2+5** **26**	All 8	3 18	5 42
TS $k = 20$	**All** **7**	5 37	**All** **5**	**All** **11**	**2** **18**

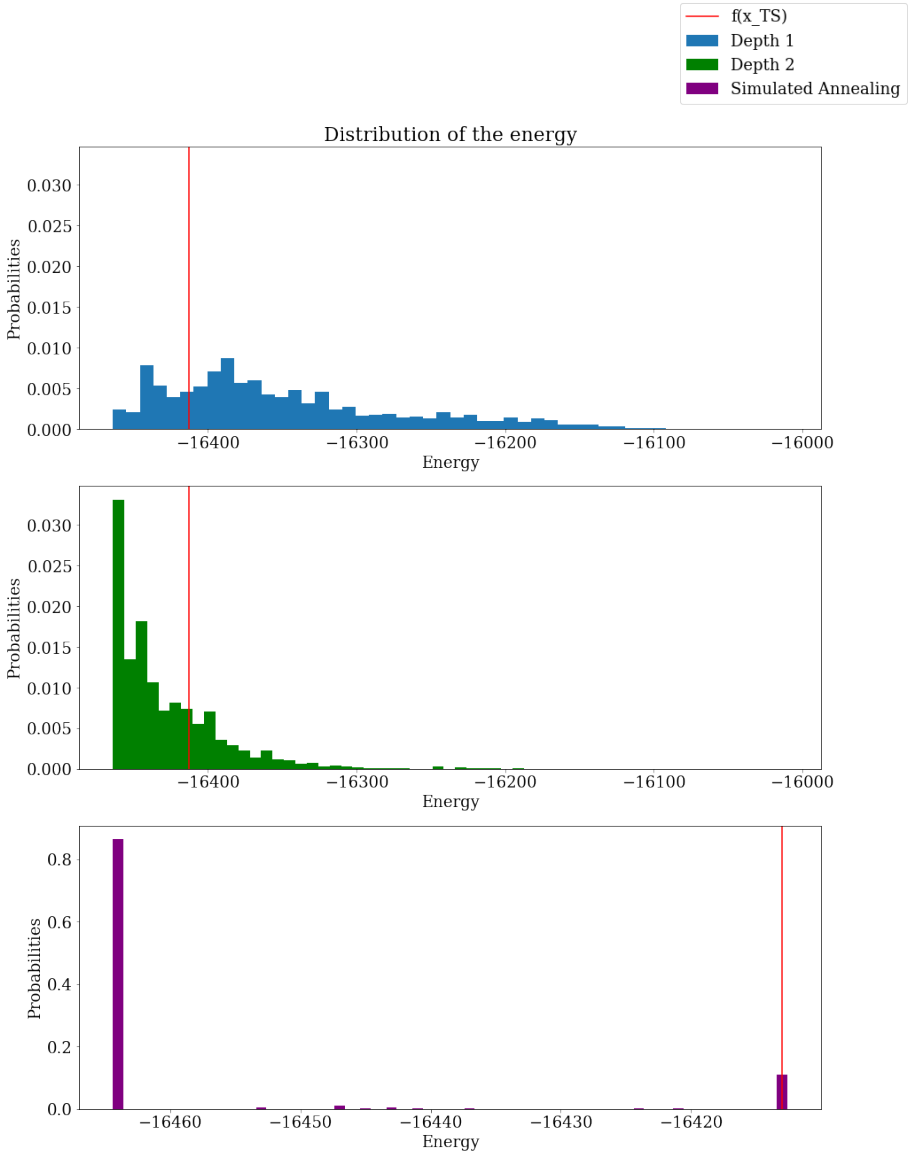

Fig. 4. QUBO evaluation distribution given by sampling 100000 times from the distribution given by the QAOA quantum state for the last TS iteration done on instance 1e. We get also the distribution from running simulated annealing 1000 times with a low temperature, that when increased always yield the optimum.

Table 2. Best evaluations obtained with the first TS iteration to reach the corresponding maximum given in [21] on instances 1d to 5d. Basic TS corresponds to running Alg.1 for 20000 iterations. The other algorithms are run for 1000 iterations. The brute-force approach with $k = 20$ is run for 200 iterations. The best performances per instance and tabu tenure are highlighted in bold.

Algorithm	TT	1d (6333)	2d (6579)	3d (9261)	4d (10727)	5d (11626)
Basic TS	5	6272 64	6579 71	9261 90	10727 103	11613 109
TS $k = 10$		6328 13	6579 160	9261 73	10727 43	**11626** **19**
TS $k = 15$		6333 10	**6579** **26**	9261 8	10727 22	11626 42
TS $k = 20$		**6333** **7**	6579 37	**9261** **5**	**10727** **11**	11613 9
Basic TS	10	6333 658	**6579** **71**	9261 485	10714 91	11626 149
TS $k = 10$		6333 568	6579 121	9247 294	10727 72	**11626** **19**
TS $k = 15$		6333 10	6579 287	9261 8	10714 685	11626 988
TS $k = 20$		**6333** **7**	6531 149	**9261** **5**	**10727** **11**	11613 9
Basic TS	15	6333 300	**6579** **71**	9261 750	10727 545	11626 891
TS $k = 10$		6328 13	6579 560	9261 422	10684 822	**11626** **19**
TS $k = 15$		6333 10	6533 12	9261 8	10686 545	11589 7
TS $k = 20$		**6333** **7**	6484 141	**9261** **5**	**10727** **11**	11613 9

Table 3. Best evaluations obtained with the first TS iteration to reach the corresponding maximum given in [21] on instances 1e to 5e. The best performances per instance and tabu tenure are highlighted in bold.

Algorithm	TT	1e (16464)	2e (23395)	3e (25243)	4e (35594)	5e (35154)
Basic TS	5	16410 132	23329 199	25228 289	35578 147	35126 219
TS $k = 10$	5	16439 19	23330 138	25243 21	35594 111	**35154** **25**
TS $k = 15$	5	**16464** **13**	**23395** **115**	**25243** **13**	35578 66	35154 58
TS $k = 20$	5	16458 85	23330 12	25236 54	**35594** **69**	35154 135
Basic TS	10	16458 215	23323 144	25243 190	35578 147	35149 1068
TS $k = 10$	10	16458 323	23395 254	25243 21	**35594** **122**	**35154** **25**
TS $k = 15$	10	**16464** **13**	**23395** **34**	**25243** **13**	35579 304	35154 226
TS $k = 20$	10	16425 176	23372 104	25243 146	35578 12	35102 100
Basic TS	15	16464 4107	**23395** **4253**	25243 500	**35594** **170**	35126 2683
TS $k = 10$	15	16439 19	23381 897	25243 21	35538 348	**35154** **25**
TS $k = 15$	15	**16464** **13**	23389 788	**25243** **13**	35559 795	35126 305
TS $k = 20$	15	16415 187	23330 12	25124 140	35578 12	34986 33

Table 4. Best values of tabu tenure (ranging from 2 to 10, adding 15) achieving the optimum obtained with the first TS iteration to reach the corresponding maximum given in [21] on instances 1e to 5e. The best performances per instance are highlighted in bold. The mention All means all TT values reached the same solution. $6+7$ means $TT = 6$ and $TT = 7$. The NA mention means no run returns the optimum, with the best value obtained in parenthesis.

Algorithm	1e (16464)	2e (23395)	3e (25243)	4e (35594)	5e (35154)
Basic TS	9 419	6 493	10 190	15 170	6 238
TS $k = 10$	8 485	7 111	All 21	6 55	All 25
TS $k = 15$	**All** **13**	**6+7** **31**	**All** **13**	3 41	**2** **20**
TS $k = 20$	2 35	NA (23370) 82	3 44	**3** **25**	4 31

Table 5. Best values of tabu tenure (ranging from 2 to 10, adding 15) achieving the optimum obtained with the first TS iteration to reach the corresponding maximum given in [21] on instances 1f to 5f. The best performances per instance are highlighted in bold. If no run returns the optimum, we give the best value achieved and in parenthesis the corresponding TT value. The NA mention means no run returns the optimum, with the best value obtained in parenthesis.

Algorithm	1f (61194)	2f (100161)	3f (138035)	4f (172771)	5f (190507)
Basic TS	8 970	7 795	4 449	172734 (15) 1148	190502 (6) 647
TS $k = 10$	**15** **213**	7 337	15 77	172449 (4) 110	190502 (5) 126
TS $k = 15$	8 225	**8** **173**	15 139	172734 (All) 46	190502 (9) 383
TS $k = 20$	NA (61087) 184	NA (100158) 101	**4** **57**	**4** **57**	**9** **150**

Table 6. Best evaluations obtained with the first TS iteration to reach given in [21] on instances 1f to 5f. The best performances per instance and tabu tenure are highlighted in bold.

Algorithm	TT	1f (61194)	2f (100161)	3f (138035)	4f (172771)	5f (190507)
Basic TS		60770 406	99959 498	138031 464	172449 510	190498 510
TS $k = 10$	5	60980 92	100047 228	138035 183	172449 127	190502 126
TS $k = 15$		60988 43	**100161** **272**	138021 697	172734 46	190406 90
TS $k = 20$		**61057** **179**	100158 101	**138035** **61**	**172748** **186**	**190502** **74**
Basic TS		61189 746	99850 1317	138035 631	172391 834	190498 720
TS $k = 10$	10	61087 310	**100150** **564**	137988 83	172346 697	190445 587
TS $k = 15$		**61194** **444**	100136 543	**138035** **139**	**172734** **46**	**190502** **593**
TS $k = 20$		61087 184	99788 177	137967 112	171991 120	190373 117
Basic TS		61193 1450	**100161** **5337**	138035 927	172734 1148	**190502** **2045**
TS $k = 10$	15	**61194** **213**	100120 579	**138035** **77**	172413 86	190320 796
TS $k = 15$		61178 998	100043 477	137890 912	**172734** **46**	190374 517
TS $k = 20$		60947 90	100051 179	137657 134	172309 190	189588 26

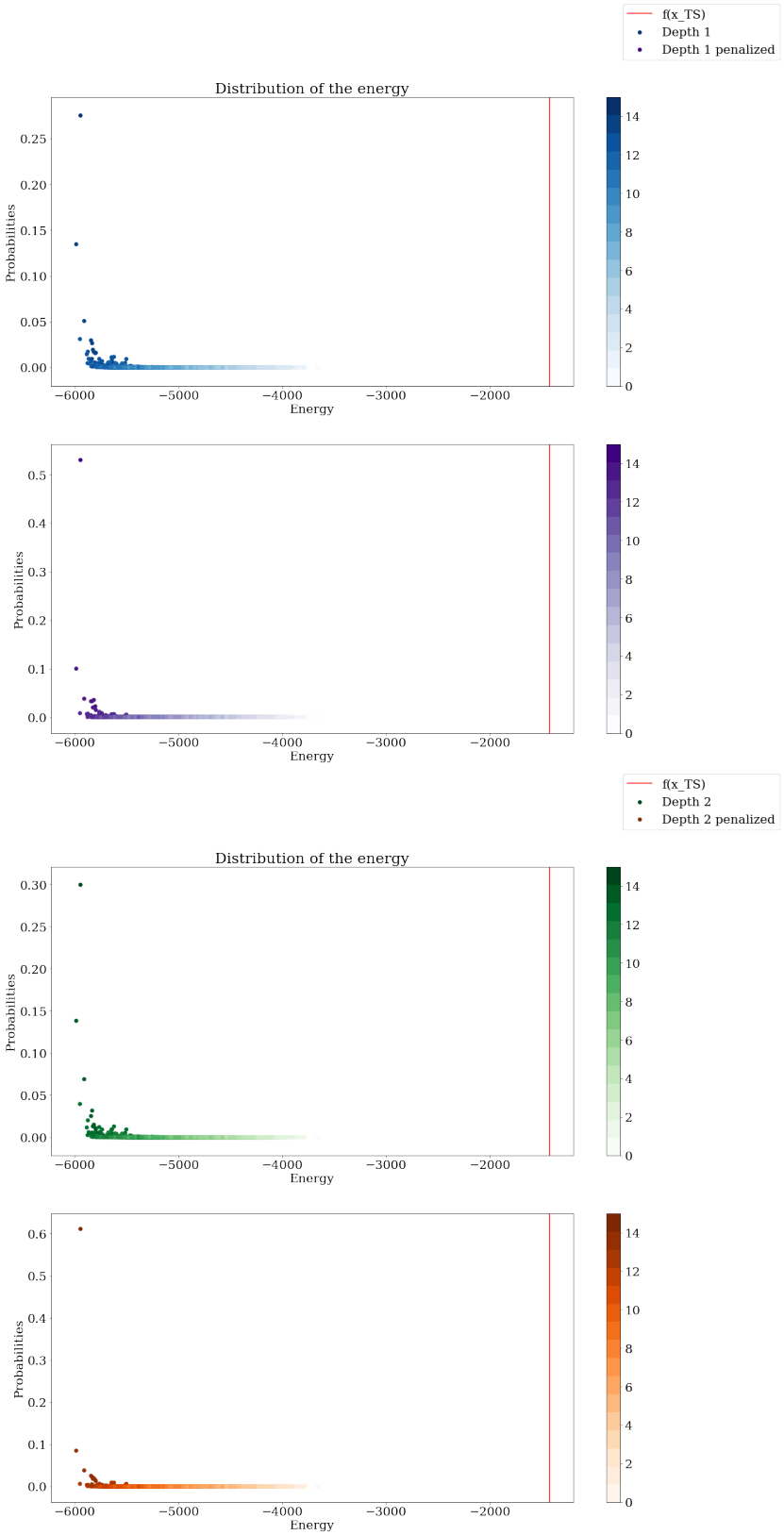

Fig. 5. Distribution of the evaluations (or energies) for the third iteration obtained on instance 1e, given by the QAOA output state at depth 1 and 2, with and without the penalty term. A colormap is given for the Hamming distance with the current tabu solution.

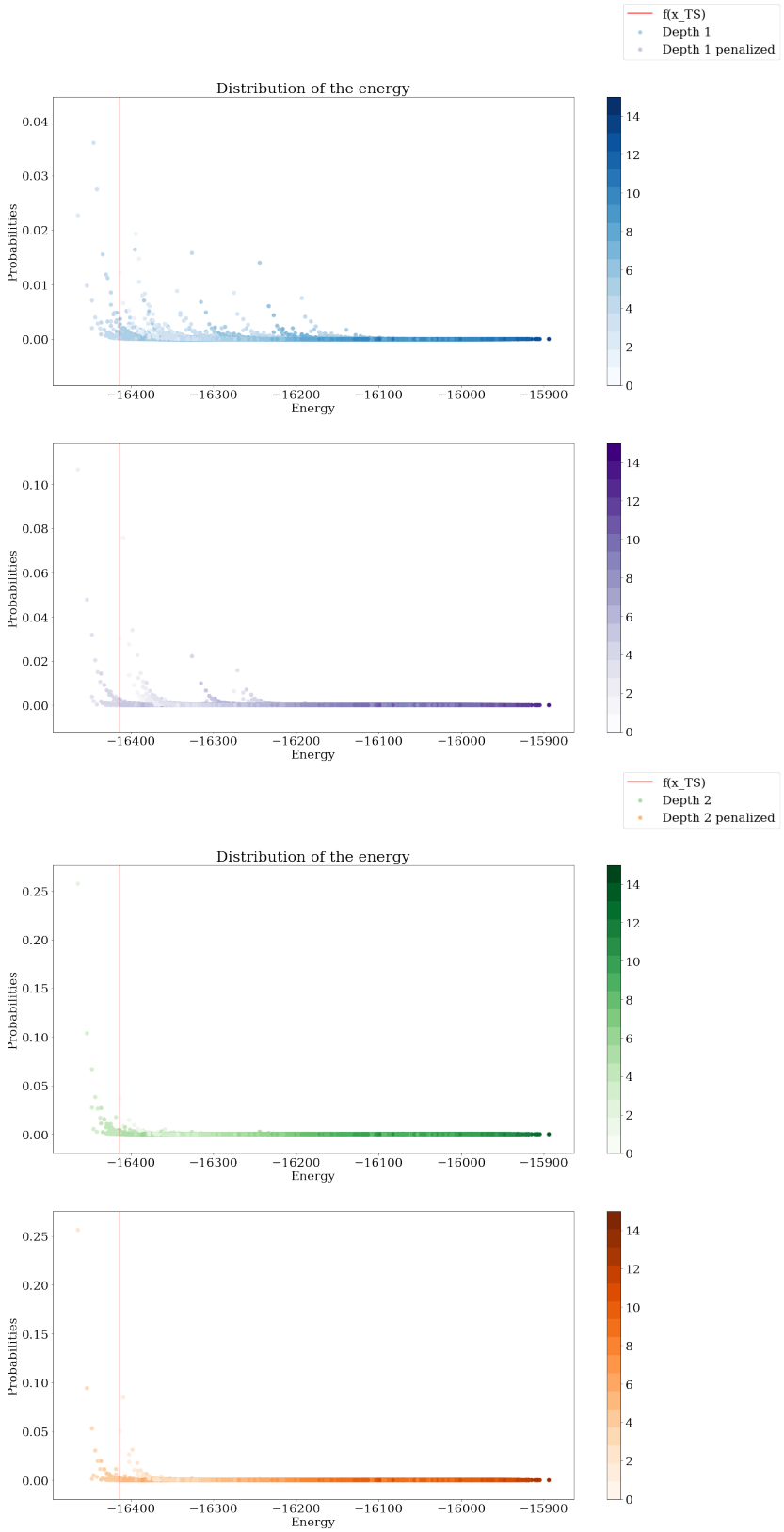

Fig. 6. Distribution of the evaluations (or energies) for the last iteration obtained on instance 1e, given by the QAOA output state at depth 1 and 2, with and without the penalty term. A colormap is given for the Hamming distance with the current tabu solution.

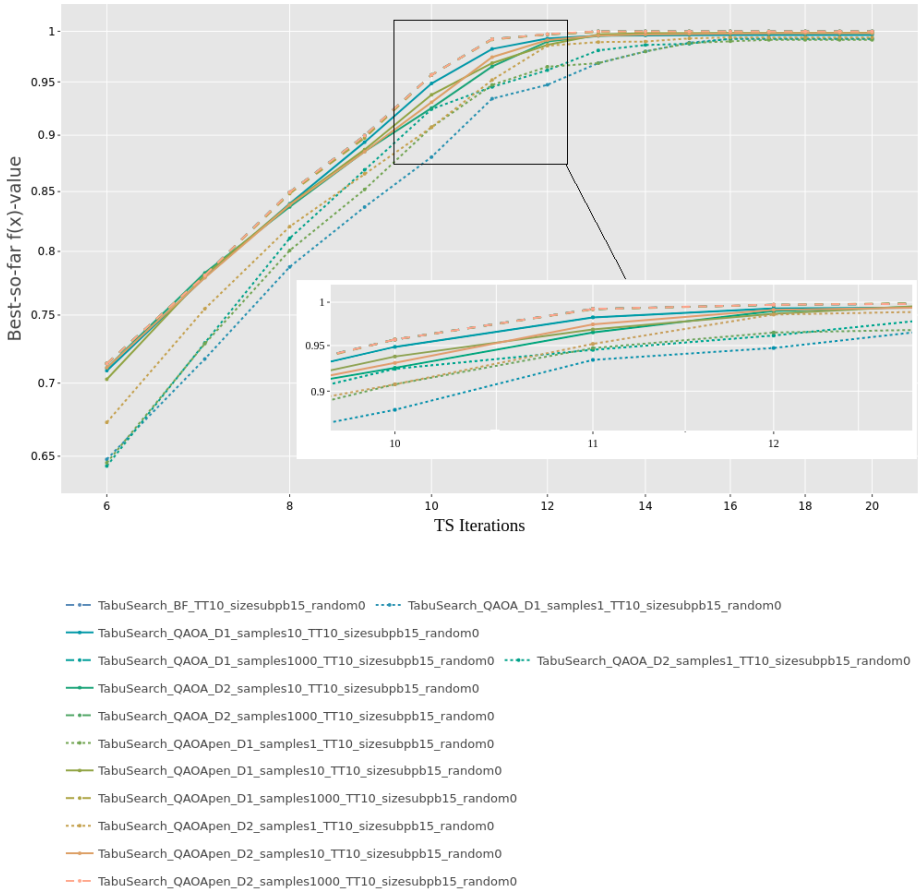

Fig. 7. Median of best normalized evaluations achieved over TS iteration for instance 1e. The ≪BF ≫mention means brute-force generation while D1, D2, mean respectively $p = 1, 2$ and the penalized version is indicated by ≪pen ≫. A higher curve corresponds better solutions reached by TS iteration. At iteration 20, the basic TS value would be 0.19, while the lowest QAOA curve value is 0.991. The 1000 samples runs and BF are over the 0.99 starting at the 11th iteration. At iteration 7, the original depth 2 QAOA using 10 samples point is higher than the others, even BF (0.7827 vs 0.7808). This also happens at iteration 12, with the penalized $p = 2, m = 1000$ (0.997085 vs 0.996902).

Table 7. Frequency of successful runs in beating the basic TS and finding the optima for instance 1d for $k = 15$ and $TT = 5$, in terms of QAOA settings (with the number of measurements noted m). We report also the number of iterations that led to the optimum 6333.

QAOA version	m	Frequency beating TS (/10)	Frequency optimum (/10)	Iterations to optimum
D1		0	0	
D1pen	1	0	0	
D2		0	0	
D2pen		0	0	
D1		7	1	18
D1pen	10	4	0	
D2		5	2	11
D2pen		4	1	10
D1		8	8	9, 10, 12
D1pen	1000	8	8	10
D2		9	9	10, 11
D2pen		10	10	9,10

Table 8. Frequency of successful runs in beating the basic TS and finding the optima for instance 1e for $k = 15$ and $TT = 10$, in terms of QAOA settings (with the number of measurements noted m). We report also the number of iterations that led to the optimum 16464.

QAOA version	m	Frequency beating TS (/10)	Frequency optimum (/10)	Iterations to optimum
D1		0	0	
D1pen	1	0	0	
D2		1	1	14
D2pen		0	0	
D1		2	2	15, 16
D1pen	10	2	2	15, 15
D2		1	1	14
D2pen		0	0	
D1		7	7	13
D1pen	1000	5	5	12, 13
D2		9	9	13, 14
D2pen		7	7	12,13