LE-QAOA: A Complexity Theory Enhanced Tool for Quantum Optimization

Carla Piazza
Department of Mathematics, Computer Science and Physics
Univerity of Udine
Udine, Italy
carla.piazza@uniud.it

Riccardo Romanello
Department of Mathematics, Computer Science and Physics
Univerity of Udine
Udine, Italy
riccardo.romanello@uniud.it

Beniamino Todone
Department of Mathematics, Computer Science and Physics
Univerity of Udine
Udine, Italy
todone.beniamino@spes.uniud.it

***Abstract*—Combinatorial optimization problems are at the heart of modern computational challenges; however, their NP-hard nature often makes exact classical solutions intractable. The Quantum Approximate Optimization Algorithm (QAOA) has emerged as a promising candidate for NISQ-era optimization. However, a significant performance gap exists between unconstrained and constrained formulations. The former refers to problems in which each solution is valid, either bad or good. The latter, to tasks in which there exist illegal solutions. While native constrained QAOA mixers offer high solution quality, they often suffer from prohibitive runtimes. In this paper, we introduce LE-QAOA (L-reduction Enhanced QAOA), a framework that leverages classical complexity theory to bridge this gap. By utilizing L-reductions to map diverse problems into a canonical MAX2SAT form, we are able to solve constrained problems using efficient unconstrained subroutines followed by a classical repairing tournament. Our results on the Maximum Independent Set (MIS) problem show that LE-QAOA achieves solution quality that matches and frequently exceeds that of native constrained solvers, while requiring as little as 10% of the execution time. This approach effectively combines the scalability of unconstrained quantum evolution with the precision of constrained optimization. Moreover, this approach makes it possible to solve combinatorial optimization by reduction to a single chosen problem whose QAOA encoding is optimized, rather than explicitly building specific Hamiltonians.**

***Index Terms*—QAOA, L-reduction, Maximum Independent Set, Constrained Optimization, Approximability**

INTRODUCTION

Combinatorial optimization problems are ubiquitous in fields ranging from logistics [1] to drug discovery [2], yet they are notoriously difficult to solve due to their NP-hard complexity. For decades, classical complexity theory has established a rigorous framework of reductions and completeness that allows us to understand the relative difficulty of these tasks [3]. With the advent of the NISQ era, the Quantum Approximate Optimization Algorithm (QAOA) has been proposed as a hybrid classical-quantum method for solving these tasks [4]. However, a major practical challenge has emerged in distinguishing between constrained and unconstrained problems. These two terms refer to combinatorial optimization problems whose search spaces are either partitioned into legal and illegal solutions, or remain unpartitioned, respectively.

For example, the MAXIMUM INDEPENDENT SET (MIS) is a constrained problem. Given a connected undirected graph $G = (V, E)$, not every subset $V' \subseteq V$ constitutes a legal independent set. Conversely, in unconstrained problems such as MAX2SAT, every possible truth assignment is considered legal. In such cases, the quality of a solution is measured solely by the number of clauses the assignment satisfies.

Due to the fundamental nature of the Quantum Approximate Optimization Algorithm (QAOA), solving problems from the latter category is significantly more straightforward, as the search space does not require restriction. Consequently, the search is driven exclusively by the solution cost—where higher values indicate superior solutions. On the other hand, fully leveraging QAOA for the former category remains an ongoing research challenge [5]–[9]. Among the various approaches adopted, we highlight two of them. The former involves incorporating penalty terms for illegal solutions within the encoded cost functions. However, the quality of the solution may decrease. The latter approach is built around the idea of adding extra *hard* constraints to the problem, to force the search to be done only on legal solutions. The drawback of this method is that of drastically increasing the execution time of QAOA circuits [10].

In this paper, we propose a novel perspective to bridge the gap between constrained and unconstrained solvers for QAOA. Our approach does not rely on modifying the Hamiltonians provided to the algorithm; rather, it utilizes pre- and post-processing techniques derived directly from classical complexity theory. Since its inception, complexity theory has utilized *reductions* to analyze the hardness of NP-problems. While

This work has been partially supported by INdAM-GNCS project *Algebra lineare quantistica, state preparation e compilazione di circuiti quantistici* (CUP E53C25002010001) and by the regional project QUASAR-FVG *Calcolo e simulazione quantistica: sviluppo, applicaizoni e ricerca in Friuli Venezia Giulia* (CUP G23C25001510002).

reductions are typically employed in their *decisional* versions (mapping yes/no instances of one problem to yes/no instances of another), a different toolset is required for optimization problems.

To achieve this endeavor, we adopt the *L-reduction* framework, first introduced in [11]. Unlike classical reductions, which are simple mappings between problems, an L-reduction must ensure that a high-quality solution for the source problem maps to a *comparably* high-quality solution for the target problem. Thus, the preservation of solution quality must be predictable. As L-reductions remain computable in polynomial time, our objective is to map constrained NP-hard problems, such as MIS, to unconstrained ones, such as MAX2SAT. To this end, we introduce LE-QAOA, a complexity-enhanced tool for quantum optimization. The framework operates through a three-stage pipeline: (*i*) Encode, where an input problem is transformed via L-reduction into a canonical MAX2SAT Ising Hamiltonian; (*ii*) Solve, where a high-speed unconstrained QAOA routine samples candidate solutions; and (*iii*) Repair, where a classical tournament projects these candidates back into the valid subspace. We evaluated our framework on the MIS problem, comparing it with both unconstrained and constrained native baselines. Our empirical data reveal that LE-QAOA consistently produces solutions of higher quality than the unconstrained baseline while matching or exceeding the constrained native solver. On top of that, we achieved these high-fidelity results while utilizing approximately 10% of the runtime required by constrained mixers, effectively tracking the scalability of unconstrained solvers. The remainder of this paper is organized as follows. Section I provides the theoretical preliminaries regarding L-reductions and the standard QAOA formulation. Section II proves the theoretical results required to understand the role of L-reductions in QAOA. Section III details the LE-QAOA architecture and specific Hamiltonian encodings used. Section IV presents our experimental setup and comparative performance analysis. Finally, Section V concludes the paper and discusses the future research directions.

I. PRELIMINARIES

The goal of this section is to introduce the reader to all the required theoretical background for a full comprehension of the paper. First, we will present all the notions about complexity theory, reductions and approximability theory. Subsequently, a brief introduction to Quantum Approximate Optimization Algorithm (QAOA) will be given.

We refer the reader to [3], [4], [11] for a more in-depth description of complexity theory, approximability theory and QAOA, respectively.

A. Complexity and Approximability Theory

Let A and B be two functional problems. We say that A reduces to B (in symbols, $A \preccurlyeq B$) if there exist two functions, R and S, both computable in logarithmic space, such that:

- if x is an instance of A, then $R(x)$ is an instance of B;
- if z is the correct solution of B, then $S(z)$ is the correct solution of A.

Intuitively, if A reduces to B, we can say that B is, in some way, more complex than A, since any efficient algorithm capable of solving B immediately results in an efficient algorithm to solve A. Given a complexity class $\mathcal{C}$, we say that a problem $A \in \mathcal{C}$ is $\mathcal{C}$-complete if:

$$\forall B(B \in \mathcal{C} \implies B \preccurlyeq A)$$

Notice that each pair (A, B) of $\mathcal{C}$-complete problems is such that $A \preccurlyeq B$ and $B \preccurlyeq A$. Then, instead of having to search for a good algorithm for every possible C-complete problem, we can choose to focus all of our efforts on optimizing the strategy to solve one $\mathcal{C}$-complete problem, and, thanks to reductions, find in this way a good solution for all the other problems in the class.

For the class NP, the problem that took this central position is SAT. Let Φ be a boolean formula in conjunctive normal form (CNF). SAT requires to decide if there exists a truth assignment of the variables that satisfies Φ.

For most NP problems, the best solution known is to reduce them to SAT and then use a SAT-solver, a tool specialized in solving instances of SAT [12].

An optimization problem A is a functional problem with the following added properties: for every possible input x, we have a set $F(x)$ which represents the set of feasible solutions for instance x; for every feasible solution $z \in F(x)$, we have a positive cost $c(z)$; the optimum on some input x is the maximum cost (or minimum cost, if we are talking about minimization instead of maximization) of any feasible solution:

$$OPT(x) = \max_{z \in F(x)} c(z)$$

and finally, we say that a solution $z \in F(x)$ is optimal if $c(z) = OPT(x)$.

In some practical cases, we may not require to find the optimal solution to a problem (usually too costly to compute), since a good approximation of the optimum can often be enough. We then focus our attention on approximation algorithms.

Definition I.1 (ϵ-approximation)**.** *Given an optimization problem G and an algorithm M such that $\forall x(M(x) \in F(x))$, we say that M is an ϵ-approximation algorithm (with $\epsilon \geq 0$) if, for every x:*

$$\frac{|OPT(x) - c(M(x))|}{max\{OPT(x), c(M(x))\}} \leq \epsilon$$

We are interested in approximation algorithms that require at most polynomial time, since they would bring an effective advantage to NP-hard problems. For this reason, it is useful to define the notion of approximation threshold:

Definition I.2 (Approximation Threshold)**.** *Given an optimization problem G, the approximation threshold of G is*

the smallest ϵ for which there exists a polynomial time ϵ-approximation.

The smaller the approximation threshold, the easier it is to approximate the problem. In particular, an approximation threshold of 1 indicates total inapproximability: finding any approximation is as hard as finding the optimum. On the other hand, an approximation threshold of 0 (meaning that for every $\epsilon > 0$ there is a polynomial time ϵ-approximation) represents the possibility of approximating the problem to an arbitrary degree; we say that problems like these have a Polynomial Time Approximation Scheme (*PTAS*). It is interesting to notice how problems that in the decisional case are all NP-complete behave differently in the context of approximation: many of these have an approximation threshold between 0 and 1; some, like TSP and INDEPENDENT SET, cannot have an approximation threshold smaller than 1 unless P = NP [11], and others, like *Knapsack*, have a *PTAS*.

Let A and B be two optimization problems such that $A \preccurlyeq B$. Assume there exists a polynomial time ϵ-approximation for B. What we ask ourselves is whether such algorithm is an approximation one for A as well. The answer is no, since the function S used in the reduction does not guarantee the quality of approximation to be preserved. For this very reason, in [11], a new kind of reduction was introduced:

Definition I.3 (L-reduction)**.** *Given two optimization problems A and B, an L-reduction from A to B ($A \preccurlyeq_L B$) is a pair of functions R and S such that:*

- *if x is an instance of A, then $R(x)$ is an instance of B;*
- *if $z \in F(R(x))$, then $S(z) \in F(x)$;*
- *it holds that:*

$$OPT(R(x)) \leq \alpha \cdot OPT(x)$$

  for some constant α;
- *for every $z \in F(R(x))$, it holds that:*

$$|OPT(x) - c(S(z))| \leq \beta \cdot |OPT(R(x)) - c(z)|$$

  for some constant β.

Like in classical reductions, it is required that both R and S are computable in logarithmic space, although in practice a polynomial time complexity is usually more than enough. L-reductions guarantee that the solution $S(z)$ found for A can not be much worse than the one found for B. In fact, it can be shown that if $A \preccurlyeq_L B$ and B has an ϵ-approximation, then A has an ϵ'-approximation, with $\epsilon' = \frac{\alpha\beta\epsilon}{1-\epsilon}$. This is not always enough to show that A has an approximation threshold lower than 1, even if problem B does. However, if B has a *PTAS* then it immediately follows that A has a *PTAS* as well. The introduction of this new type of reduction was the trigger for the creation of a new complexity class, called MaxSNP.

Such class, however, is built on top of a smaller one called MaxSNP_0, which is defined as follows:

Definition I.4 (MaxSNP_0)**.** *A problem is in MaxSNP_0 if it can be written as:*

$$\max_S |\{(x_1, ..., x_r) \in V^r : \Phi(G, S, x_1, ..., x_r)\}|$$

where G is a relation describing the input and V is a domain depending on the input.

The problems inside this new class serve as a *core* for the ones in MaxSNP.

Definition I.5 (MaxSNP)**.** *A problem A is in* MaxSNP *if it can be L-reduced to a problem in MaxSNP_0.*

An example is the problem MAX-CUT. Let $G = (V, E)$ be an undirected graph. The problem requires to find the cut $S \subseteq V$ that maximizes the number of edges (u, v) for which $u \in S$ and $v \notin S$.

We show that this problem belongs to MaxSNP_0 by writing it as:

$$\max_{S \subseteq V} |\{(x, y) : G(x, y) \wedge S(x) \wedge \neg S(y)\}|$$

where, V is the set of nodes of the graph, S is a unary relation that represents a subset of nodes, and G is the binary relation that represents the input graph ($G(x, y)$ holds iff the nodes x and y are adjacent). This means that we are searching for the subset S of nodes that maximizes the number of edges crossing the cut, i.e. the maximum cut.

We say that a problem A in MaxSNP is MaxSNP-complete if every problem in MaxSNP L-reduces to A. In [11], many problems are shown to be MaxSNP-complete; some interesting ones are MAX-CUT, MAX3SAT, MAX2SAT and K-DEGREE INDEPENDENT SET.

In [13], authors have shown that if there exists a MaxSNP-complete problem with a *PTAS*, then we could conclude that P = NP. While this seems a negative result, it can still give us some insight on the nature of approximation problems: *PTAS*'s seem to play the role that in the decisional case was held by polynomial time algorithms, and MaxSNP seems to be the adequate approximation counterpart of NP.

It is also possible to define some extensions of MaxSNP. A particularly useful one is the class that contains the weighted versions of the problems in MaxSNP like WEIGHTED MAX-CUT and WEIGHTED MAX2SAT. We will refer to this class as W-MaxSNP, for Weighted MaxSNP. Most of the results found for MaxSNP hold for W-MaxSNP as well.

B. QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm used to find approximate solutions to combinatorial optimization problems, first introduced in [4]. It uses n qubits, where the 2^n possible configurations that can result from a measurement represent the search space (the set of possible solutions to our combinatorial problem). The initial state of these qubits is evolved applying in alternation two operators, described in terms of their Hamiltonians:

- the cost Hamiltonian H_C, which encodes in the phase the information about the cost of each state. We can describe the general behavior of H_C as:

$$H_C = \sum_{x \in \{0,1\}^n} c(x)|x\rangle\langle x|$$

- the mixing Hamiltonian H_M, which allows to explore the search space. The most common H_M used is the bit flip operator:

$$H_M = \sum_{i=1}^{n} X_i$$

Intuitively, we can see the effect of an application of H_M as a single step in a graph where there is a node for each state in the search space, and an edge between two nodes x and y if the bit strings representing states x and y differ in only one bit (if H_M is the bit flip operator). Multiple applications of H_C and H_M can thus be seen as a quantum walk over the search space graph, attracted by the states with a high cost function. Because of the rules of quantum computation, a quantum state can evolve just via unitary operations. Therefore, H_C and H_M cannot be applied directly to a quantum state. However, they are both Hermitian operators ($H_C = H_C^\dagger$ and $H_M = H_M^\dagger$). Therefore, we can use the following equalities:

$$U_C = e^{i\gamma H_C}$$
$$U_M = e^{i\beta H_M}$$

to obtain unitary operations U_C and U_M, for some real values γ and β. States associated to a high cost function, when evolved with these two operators, will see their phase amplified, and will thus be the result of a measurement with high probability. We call a single application of U_C and U_M a layer, and we use p to denote the number of layers applied. Each layer uses a different pair of γ and β; thus we describe these parameters as two lists of angles $\gamma = (\gamma_1, \gamma_2, ..., \gamma_p)$ and $\beta = (\beta_1, \beta_2, ..., \beta_p)$. These lists of angles are usually computed classically, for example with some method like the gradient descent [14]. Let $M_p(x)$ be the solution computed by QAOA with p layers on some input x. It was proven in [4] that:

$$\lim_{p \to \infty} \mathbb{E}[c(M_p(x))] = OPT(x) \tag{1}$$

where $\mathbb{E}$ denotes the expected value. Such result allows us to say that if p is allowed to grow enough, then the solution computed by QAOA tend to be optimal ones.

If a problem is such that, for any input, the feasible subspace is equal to the search space, then the initial state $|s\rangle$ can be set to be the superposition of all the 2^n states:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

We call a problem like this an *unconstrained problem*. For example, MAX-CUT is unconstrained, since any subset of nodes is a legal cut. On the other hand, if a problem has some solutions in the search space that are not feasible, we say it is *constrained*. INDEPENDENT SET belongs to this category, since not every subset of nodes in a graph is an independent set. Encoding constrained problems in QAOA is a bit more troublesome than the unconstrained case. There are two main possibilities:

- Unconstrained encoding: we treat the problem as if it was unconstrained (every solution in the search space is legal), but we give a penalty to the cost function for every hard constrained that is not satisfied. In this way, the probability of measuring an unfeasible state is very low. This method has the advantage of being very simple: like the unconstrained case, the initial state can be chosen to be the very convenient superposition of all states, and H_M is the simple bit flip operator. However, there is a non-zero probability that the measure leads to a non feasible result.
- Constrained encoding [6]: we use as initial state a single state that we know to be feasible, and build H_M in a way that, when applied to a feasible state, it can never evolve to an unfeasible one. In this way, we can never end up outside the feasible subspace. In this case, the mixing Hamiltonian is way more complex than the very simple bit flip operator. Moreover, it is much harder to use a superposition of states as the initial state, since we cannot include in the superposition no unfeasible state. The most prominent drawback of this approach is a decrease in efficiency and an increase of the circuits' depth.

An example of a particularly demanding constrained problem is KNAPSACK, since the best known way to encode its hard constraints requires the use of many extra qubits, leading to a growth in the size of the search space [7].

QAOA can be interpreted as a discrete version of the Quantum Adiabatic Algorithm (QAA). QAA is based on the Adiabatic theorem, which states that a physical system remains in its current eigenstate if a perturbation acts on in slowly enough. In other words, if a system with Hamiltonian H is in an eigenstate $|s\rangle$ and a perturbation acts on it modifying H, if the perturbation is slow enough, then the resulting system, with Hamiltonian H', will be in the eigenstate $|s'\rangle$ corresponding to $|s\rangle$ (for example, if $|s\rangle$ is the ground state of H, then $|s'\rangle$ is the ground state of H').

The idea of QAA is thus to build a problem Hamiltonian H_P in such a way that its ground state encodes the correct solution to our problem. This Hamiltonian could however be very complex, and its ground state too hard to compute. We then create another Hamiltonian, called the initial Hamiltonian H_B, chosen so that it is easy to prepare the system in its ground state. At this point, we evolve adiabatically H_B into H_P. The resulting system will be described by this Hamiltonian:

$$H(t) = \left(\frac{1-t}{T}\right) H_B + \left(\frac{t}{T}\right) H_P$$

where T is the total time of evolution of the system and t is the current time (going from 0 to T). Since H_B was initially in its ground state, thanks to the adiabatic theorem, the system at the end will be in the ground state of H_P, which is the solution which we were searching for.

In QAOA, H_M takes the role of the initial Hamiltonian H_B, while H_C that of H_P. The main difference here is that the evolution is not continuous, but done in steps, which correspond to the alternate applications of U_C and U_M. For p tending to infinity, the effect of QAOA tends to that of QAA, so p being finite is what makes QAOA an *approximate* algorithm. However, this discretization is one of the main advantages of QAOA over QAA, since in the latter case the required time of evolution could be, for some problems, very high, while in QAOA, by choosing the parameter p, we can balance the quality of the approximation with the time of execution (although QAOA has already been shown to be effective even for p equal to 1 or 2). Moreover, notice that QAOA is quite easy to implement on a quantum circuit architecture, while QAA needs a completely different kind of quantum computer.

II. L-REDUCTIONS IN QAOA

QAOA is, in general, not an ϵ-approximation algorithm, since it does not guarantee an upper threshold to the relative error of the solution. However, in this section we will show that L-reductions are still very useful in the context of this quantum algorithm. We will start pointing out a simple result:

Proposition 1. *Given two maximization problems A and B such that $A \preccurlyeq_L B$ and an input instance x of A, if z is a feasible solution of $R(x)$, then it holds that:*

$$\frac{c(S(z))}{OPT(x)} \geq \alpha\beta \left(\frac{c(z)}{OPT(R(x))} - 1 \right) + 1$$

The quantity $\frac{c(z)}{OPT(R(x))}$ is the approximation ratio of solution z for problem B, while $\frac{c(S(z))}{OPT(x)}$ is the approximation ratio of the solution found via L-reduction for problem A, to which this inequality poses a lower bound.

Proof. From the properties of L-reductions, we know that:

$$OPT(x) - c(S(z)) \leq \beta \cdot (OPT(R(x)) - c(z))$$

Let $\frac{c(z)}{OPT(R(x))} = r$; we can rewrite the previous inequality as:

$$OPT(x) - c(S(z)) \leq \beta \cdot OPT(R(x))(1 - r) \leq$$
$$\leq \alpha\beta \cdot OPT(x)(1 - r)$$

Thus:

$$OPT(x) - c(S(z)) \leq \alpha\beta \cdot OPT(x)(1 - r)$$

that easily yield to the thesis

□

In the case of minimization problems the result is slightly different, but the meaning is the same: we find a lower bound to the approximation quality of the solution for problem A that depends on the approximation quality of the solution found for problem B.

Recalling Equation 1, it is straightforward to obtain what follows:

Lemma II.1. *Given two optimization problems A and B such that $A \preccurlyeq_L B$ and an instance x of A, we get:*

$$\lim_{p \to \infty} E[c(S(z_p))] = OPT(x)$$

where z_p is an approximate solution computed by QAOA with p layers on input $R(x)$.

Suppose problem A is troublesome to solve using QAOA, while problem B has a really simple and convenient encoding; if $A \preccurlyeq_L B$, then we can just compute $R(x)$, use QAOA to find a solution z that is a good approximation of $OPT(R(x))$, and then return $S(z)$ as the solution for problem A; thanks to the previous result, we know that, for p that tends to infinity, the solution computed in this way will tend to the optimum. This means that, for example, any problem A in MaxSNP can easily be solved with QAOA through an L-reduction to a MaxSNP-complete problem, like MAX-CUT or MAX2SAT (and the same can be said for W-MaxSNP). Notice that this result is independent of the specific α and β that appear in the L-reduction (although, in practice, the lower they are, the better it is for the quality of approximation).

A. Example of L-reduction

We now present an example of L-reduction from K-DEGREE INDEPENDENT SET to MAX2SAT (the reduction here presented is a modification of the one showed in [11]).

K-DEGREE INDEPENDENT SET: Given an undirected graph $G = (V, E)$ where each node has degree at most k, find the largest independent set. An independent set is a collection of nodes wherein no two nodes are adjacent.

MAX2SAT: Given a set of clauses $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$, where each clause ϕ_i is a disjunction of at most two literals, find the assignment of the variables that maximizes the number of satisfied clauses.

Notice that K-DEGREE INDEPENDENT SET is constrained, while MAX2SAT is unconstrained.

We start the description of the L-reduction presenting the function R, which, given a graph $G = (V, E)$, needs to output a set of binary clauses. We create a variable x_i for each node $i \in V$; for each variable x_i, we create a unary clause (x_i); for each edge $(u, v) \in E$, we create a binary clause of the form $(\neg x_u \vee \neg x_v)$. This completes the description of function R.

We need to show that $OPT(R(x)) \leq \alpha \cdot OPT(x)$. For convenience, we set $n = |V|$ and $m = |E|$. First, notice that the number of clauses in $R(x)$ is exactly $n + m$, but, since the graph has maximum degree at most k, we have $m \leq \frac{kn}{2}$. We thus get $OPT(R(x)) \leq n(1 + k/2)$.

In a graph that has maximum degree k, it is always possible to find an independent set of size at least $\frac{n}{k+1}$: we start from a set $S = \emptyset$, at each step we add to S a node from V chosen at random, and remove it and its neighbors from the graph. At

each step we remove at most $k+1$ nodes, so we can repeat this process at least $\frac{n}{k+1}$ times, finding at the end an independent set of the stated cardinality. This means that $OPT(x) \geq \frac{n}{k+1}$; this, together with the previous result, gives us:

$$OPT(R(x)) \leq (k+1)\left(1+\frac{k}{2}\right)OPT(x)$$

so, with $\alpha = (k+1)(1+k/2)$, we have proven the first property of L-reductions.

Now we describe the function S. The main idea is simple: given a truth assignment z of $R(x)$, we consider the node i to be in the independent set if and only if variable x_i is true in z. The problem, of course, is that in this way we could get a set that is not in fact an independent set (remember that MAX2SAT is unconstrained, so every truth assignment is considered to be feasible). Then, instead of computing directly the independent set resulting from z, we first find a "repaired" truth assignment, that we call z', in this way: we check each binary clause, and, when we find one that is not satisfied (meaning that both its variables are set to *True*), we set one of the two variables to *False*. Notice that a non satisfied binary clause represent two adjacent nodes both belonging to the set, so what we are doing is simply removing one of them. Now we can use z' to find a set of nodes like we described before, and this time we know we will get an independent set.

We now need to show that $|OPT(x) - c(S(z))| \leq \beta \cdot |OPT(R(x)) - c(z)|$. Since we are talking about two maximization problems, we can remove the absolute values from the equation. Suppose that we have an assignment z for $R(x)$ that corresponds to the maximum independent set for G; if we try to set to *True* any variable x_i that in z was *False*, the number of satisfied clauses would increase by one for the clause (x_i), but would decrease by one for each binary clause in which x_i appeared together with another variable set to *True* (we always have at least one of these clauses, since otherwise z would not have been the assignment corresponding to the maximum independent set). This reasoning shows that $OPT(R(x)) = OPT(x) + m$, since z, representing a correct independent set, satisfies all the m binary clauses.

Recall now the way we compute z' starting from z: every time we set a variable x_i to *False*, we lose the clause (x_i), but we gain the binary clause that was not satisfied because of x_i being *True*, plus possibly some other binary clauses in which x_i appears. This means that $c(z') \geq c(z)$. Finally, since in z' all the m binary clauses are satisfied, it is immediate to see that $c(S(z)) = c(z') + m$. Putting all this together, we get:

$$OPT(x) - c(S(z)) = OPT(x) + m - (c(S(z)) + m) =$$
$$OPT(R(x)) - c(z') \leq OPT(R(x)) - c(z)$$

We have just proven that the second property of L-reductions is satisfied with $\beta = 1$.

The functions R and S are both computable in logarithmic space; here we do not go into the details of this proof, but notice that both functions are straightforward to compute in linear time, which, for practical purposes, is more than enough.

B. L-reductions in the QAOA literature

In [15], an unconstrained encoding is presented for MAX INDEPENDENT SET (in the version without any bound to the degree of the nodes). In this encoding, a binary variable x_i is used for each node i, where node i is taken to be part of the set if and only if $x_i = 1$. The cost function, given a graph $G = (V, E)$, is defined as follows:

$$C = \sum_{i \in V} x_i - \sum_{(u,v) \in E} x_u x_v$$

We thus try to maximize the number of nodes in the set, giving a penalty (of weight 1) for every edge that is not properly assigned (meaning that both its nodes are in the set). Now, recall the L-reduction from K-DEGREE INDEPENDENT SET to MAX2SAT we previously described; there, we count the number of nodes in the set, adding a bonus for each edge that is properly assigned (every binary clause that is satisfied). The cost function of the instance $R(x)$ of MAX2SAT can thus be described as:

$$C = \sum_{i \in V} x_i + \sum_{(u,v) \in E} (1 - x_u x_v) =$$
$$= \sum_{i \in V} x_i - \sum_{(u,v) \in E} x_u x_v + m$$

This writing exposes the similarities between our L-reduction and the penalty term approach proposed in [15]: the two methods are identical, except for an offset of m in the cost function.

With this intuition, we can build an L-reduction from MAX INDEPENDENT SET (MIS) to W-MAX2SAT$^-$.

W-MAX2SAT$^-$: Given a set of clauses $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$, where each clause is a disjunction of at most two literals, and each clause ϕ_i is assigned a weight w_i (possibly negative), find the truth assignment z of the variables that maximizes the quantity:

$$\sum_{i=1}^{n} w_i \cdot t(\phi_i, z)$$

where $t(\phi_i, z)$ is 1 if the assignment z satisfies ϕ_i and 0 otherwise.

The S function of this L-reduction is identical to the S in the L-reduction we proposed, while the R function needs a slight modification: we create all the clauses in the same way we used in the L-reduction from K-DEGREE INDEPENDENT SET to MAX2SAT, giving to each one of these clauses a weight of $+1$, but we add a new clause of the form $(x_i \vee \neg x_i)$ for some i, and give it weight $-m$. This L-reduction describes exactly the approach used in [15].

This L-reduction is in some way more powerful than the one we presented, since it is made on MIS instead of the bounded K-DEGREE INDEPENDENT SET. Furthermore, in this L-reduction both α and β are equal to 1, granting a better quality of approximation. What we lose in exchange for this power is the fact that MAX2SAT$^-$ does not belong to MaxSNP or W-MaxSNP due to its negative weights.

This complication, however, has no negative impact on the possibility to use L-reductions in QAOA to solve these problems.

This argument shows that L-reductions can be seen as a generalization of solutions that have already been used in the literature; in particular, many penalty based approaches used for constrained problems may embody some kind of L-reduction.

III. The LE-QAOA Framework

As established in the previous section, the theoretical machinery of L-reductions already implicitly underpins several existing heuristic encodings in the QAOA literature. However, leaving these connections as implicit, ad-hoc formulations overlooks the rigorous guarantees provided by classical complexity theory. Our goal is to formally expose this connection and systematically exploit the power of L-reductions within a complete, automated pipeline. By explicitly treating the transformation of constrained problems into unconstrained Hamiltonians as a mathematically grounded L-reduction, we can seamlessly delegate the burden of constraint satisfaction to classical pre- and post-processing. To achieve such endeavor, we introduce LE-QAOA (L-reduction Enhanced Quantum Approximate Optimization Algorithm), an integrated software framework designed to abstract the complexities of quantum Hamiltonian construction for combinatorial optimization. LE-QAOA is designed to be accessible to two distinct user profiles: the *expert*, who focuses on fine-tuning quantum circuits, and the *complexity practitioner*, who utilizes L-reductions to map diverse NP-hard problems onto canonical forms without requiring deep expertise in quantum mechanics.

A. Algorithmic Workflow

The LE-QAOA pipeline is structured to delegate the burden of constraint satisfaction to a polynomial-time reduction phase followed by a classical repairing tournament (the S function of L-reductions). This design philosophy treats the quantum circuit as a specialized optimization engine while utilizing classical algorithms to ensure that every result is feasible. The high-level execution flow is detailed in Algorithm 1, which illustrates the interaction between complexity-theoretic reductions and quantum heuristics.

The process begins at line 2, where the framework applies the reduction function R to the initial problem instance P. The primary objective of this stage is to transform P into a MAX2SAT formula, F, by leveraging the properties of L-reductions. This transformation is critical because it ensures that the mapping of any input problem into a quadratic Boolean form is achieved within a strictly polynomial runtime. Once the formula F is established, the framework proceeds to line 3, which is responsible for generating three distinct Hamiltonian encodings: standard, shifted, and normalized. These encodings represent the contemporary state-of-the-art for translating Boolean formulas into Ising Hamiltonians, allowing the system to investigate which specific energy landscape most effectively facilitates convergence during the optimization.

Algorithm 1 The LE-QAOA Execution Pipeline (High-Level)

```
1:  procedure LE-QAOA(P, R, S)
2:      F ← R(P)
3:      H ← GENERATEENCODINGS(F)
4:      for each Hamiltonian H ∈ H do
5:          B ← QAOA(H)
6:          for each Repairing technique S ∈ S do
7:              U ← U ∪ S(B)
8:          end for
9:      end for
10:     Return Best valid solution from U
11: end procedure
```

The subsequent phase, spanning lines 4 through 9, represents the hybrid classical-quantum core of the LE-QAOA approach. Each Hamiltonian is utilized as an input for the QAOA black box at line 5, which is executed at a specific depth p and sampled s times. By encapsulating this quantum routine as a black box, the framework allows practitioners from the complexity theory domain to harness quantum optimization without needing to manage the internal intricacies of variational parameter discovery, such as the tuning of β and γ. The output of this step is a raw bitstring $\mathcal{B}$, which serves as a candidate solution for the MAX2SAT problem.

Following the acquisition of the bitstring, the workflow shifts to the application of repairing techniques in lines 6 through 8. Because QAOA is a heuristic, the sampled strings may not inherently satisfy the hard constraints of the original problem—for instance, a bitstring may represent a set of nodes that are not independent. To resolve this, the repairing tournament $\mathcal{S}$ applies multiple heuristics to project $\mathcal{B}$ back into the valid subspace, effectively transforming a MAX2SAT result into a high-quality solution for problem P. After iterating through all encodings and repairs, the algorithm identifies and returns the highest quality valid solution stored in $\mathcal{U}$.

A defining characteristic of this framework is the strategic use of a classical backbone to support quantum optimization. By ensuring that the reduction function R is polynomial-time by definition, the complexity of the preprocessing remains low even as the problem scales. Furthermore, as long as the repairing strategies are not exact, the classical overhead remains computationally negligible. This design choice ensures that the QAOA routine remains the dominating factor in the total runtime, allowing the framework to bridge the gap between constrained and unconstrained problems without introducing significant classical bottlenecks. Ultimately, the novelty of this method lies in how it treats the QAOA routine as a compact, interchangeable black box within a broader classical complexity-theoretic architecture.

B. Formal MAX2SAT Encodings

Here, we present the Hamiltonian encodings for the specific case of MAX2SAT instances obtained with our L-reduction,

although the definitions we will give are easily generalizable to any MAX2SAT instance.

The framework implements three distinct methods for generating the cost Hamiltonian H_C. For a clause $c = (\neg x_i \vee \neg x_j)$, we utilize the spin mapping $x \to \frac{1}{2}(I - Z)$.

1) **Standard Encoding (H_{std}):** Following the canonical mapping for MAX2SAT instances [10], the Hamiltonian is the sum of local penalty operators:
$$H_{std} = \frac{1}{4} \sum_{(i,j)\in E} (I + Z_i + Z_j + Z_i Z_j)$$
2) **Shifted Encoding (H_{shft}):** To isolate the spin-spin interactions and potentially simplify the optimization landscape, the shifted encoding removes the identity factor from the standard definition [16]:
$$H_{shft} = \frac{1}{4} \sum_{(i,j)\in E} (Z_i + Z_j + Z_i Z_j)$$
3) **Normalized Encoding (H_{norm}):** This encoding introduces a weighting factor ω_{ij} to balance the influence of constraints based on node connectivity:
$$H_{norm} = \sum_{(i,j)\in E} \omega_{ij}(I + Z_i + Z_j + Z_i Z_j)$$
where $\omega_{ij} = \frac{\lambda}{\max(deg(i),deg(j))}$. Here, $deg(v)$ is the degree of variable v in the constraint graph and $\lambda \in \mathbb{R}^+$ is a scaling constant. This normalization is designed to prevent high-degree vertices from dominating the cost function, promoting more uniform convergence [5].

IV. Experimental Evaluation

To validate the efficiency and effectiveness of the LE-QAOA framework, we conducted a series of noiseless simulations targeting the Maximum Independent Set (MIS) problem. All benchmarks were executed on a local workstation equipped with an Intel Xeon W-3323 CPU (12 cores) at 3.50 GHz, 64 GB of RAM, and dual NVIDIA GeForce RTX 2080 Ti GPUs (11 GB VRAM each). This configuration was selected to ensure a reproducible experimental environment, intentionally avoiding the use of High-Performance Computing (HPC) clusters or specialized cloud-based server architectures. This choice emphasizes that the L-reduction approach democratizes quantum optimization by providing high-utility results without the need for prohibitive hardware resources.

A. Benchmarks and Baselines

The benchmarking suite was designed to test the framework across a variety of MIS instances. For each configuration defined by the number of nodes $N \in \{6, 8, 10, 12, 14, 16, 18, 20\}$ and a maximum degree constraint $d \in \{3, 4, 5, 6\}$, we generated 10 random graphs with N nodes such that the degree of each node is at most d. This ensured a diverse set of constraints for the MIS solver to navigate.

For each graph instance, the QAOA depth was set to $p \in \{1, 2\}$, and we performed circuit sampling with both $S = 500$ and $S = 1000$ shots. The performance of LE-QAOA using the Normalized encoding—which empirical data identified as the superior variant among the three proposed—was compared against two standard baselines provided by the PennyLane QAOA.MAX_INDEPENDENT_SET library:

1) Native Unconstrained MIS: A standard penalty-based QAOA implementation where MIS constraints are added to the cost Hamiltonian as soft penalties.
2) Native Constrained MIS: An MIS-specific implementation utilizing an XY-mixer to restrict the quantum evolution strictly to the valid subspace of independent sets.

B. Performance Analysis

The empirical results, summarized in Table I and visualized in Figure 1, highlight the superiority of the Normalized encoding. As shown in Figure 1a, the quality of the solutions (the average MIS size found) for LE-QAOA consistently matches or exceeds that of the Native Constrained solver.

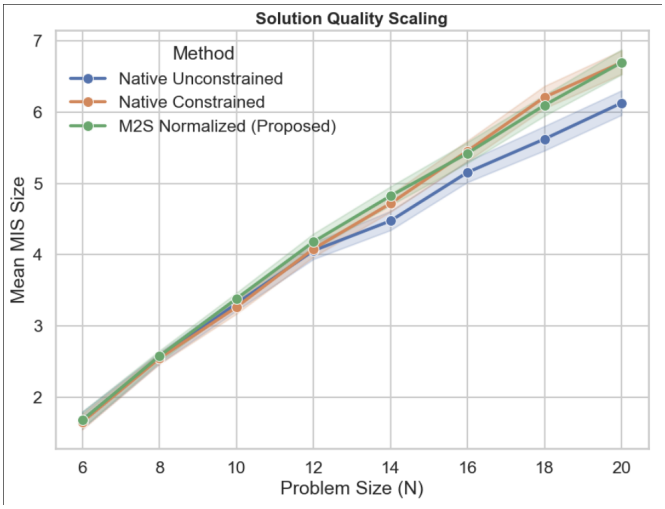
TABLE I: Average MIS Size and Runtime comparison across graph sizes. Results are averaged over all degrees, depths, and graph instances.

	LE-QAOA (Norm)		**Native Uncon.**		**Native Con.**	
N	**Avg Size**	**Time (s)**	**Avg Size**	**Time (s)**	**Avg Size**	**Time (s)**
6	1.68	0.96	1.68	0.94	1.65	4.55
8	2.58	1.24	2.55	1.19	2.54	7.75
10	3.38	1.50	3.31	1.45	3.26	9.86
12	4.18	1.79	4.06	1.71	4.08	12.07
14	4.83	2.11	4.48	2.01	4.72	14.49
16	5.43	2.55	5.16	2.44	5.46	17.58
18	6.10	3.52	5.63	3.38	6.21	25.65
20	6.70	7.10	6.13	6.95	6.70	61.99

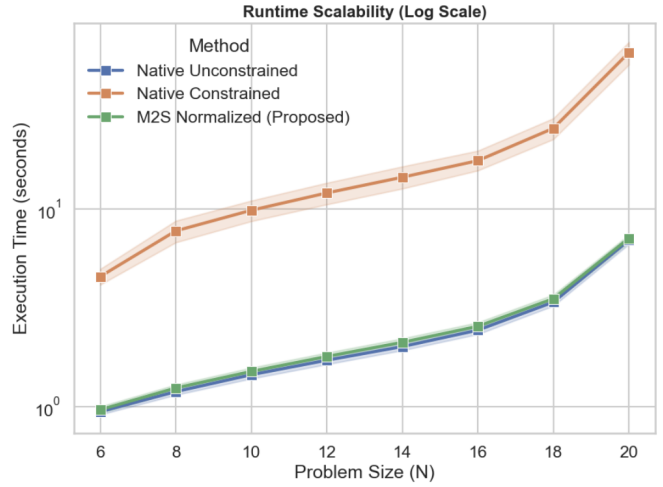
In terms of computational efficiency (Figure 1b), LE-QAOA tracks the runtime profile of the Native Unconstrained solver. While the runtime shown reflects the execution of the Normalized encoding alone, the performance margin is substantial enough to allow for significant flexibility. Our data indicates that for all the values of N, the total time required to execute all three encoding techniques (Standard, Shifted, and Normalized) is still notably less than the time required for a single execution of the Native Constrained MIS solver.

This identifies a significant "complexity-enhanced" advantage: we obtain the high-fidelity results typically reserved for complex, hardware-restricted mixers while maintaining the scalability of unconstrained solvers. It is important to highlight that while the runtime for LE-QAOA is slightly higher than that of the native unconstrained solver, this marginal increase is justified by the substantial gain in solution quality. Our empirical data shows that the quality of our solution not only matches but frequently exceeds the results obtained by the native constrained solver, particularly as the problem complexity increases.

Furthermore, since the L-reduction and the repairing techniques are polynomial-time by definition, the classical preprocessing and post-processing time remains negligible as



(a) Solution Quality (MIS Size)



(b) Execution Time (Seconds)

Fig. 1: Comparative benchmarking of LE-QAOA. The framework achieves solution quality comparable to native constrained mixers while maintaining the runtime scalability of unconstrained approaches.

N grows. In all benchmarks, the variational quantum optimization consistently remained the dominating factor in the total wall-clock time, confirming the scalability and practical utility of the approach for solving MIS and other constrained problems.

C. Code Availability

The source code of LE-QAOA is openly available at https://github.com/RiccardoRomanello/LE-QAOA. In addition to the core tool, the repository contains the data used to generate the figures shown in this work.

V. Conclusion

QAOA is a pivotal algorithm for solving optimization problems in the NISQ era. However, while solving unconstrained problems is computationally efficient, constrained instances introduce major bottlenecks, particularly regarding runtime and circuit depth. To tackle this issue, we have brought complexity theory into the quantum playground. By introducing LE-QAOA, we have demonstrated that L-reductions can be used to standardize the optimization landscape into a MAX2SAT form, allowing for the use of efficient unconstrained subroutines. Our results for MIS indicate that we can match the precision of hardware-restricted constrained mixers while maintaining the runtime scalability of unconstrained solvers. Although here we underlined the power of L-reductions in the context of constrained and unconstrained problems, it is important to notice that these results allow for the possibility of using QAOA like a SAT-solver, meaning that, instead of creating an ad-hoc Hamiltonian for each problem, we can use L-reductions to always go back to a single chosen problem (possibly W-MaxSNP-complete). The candidate we proposed to take this central position is MAX2SAT, for its simplicity and its natural connection to graph problems. Whether our candidate proves itself to be adequate, or another problem emerges as the best one, this central problem could then become the focus of all the attention in QAOA Hamiltonian optimization. Another interesting way forward could be to investigate the use of L-reductions in KNAPSACK: the encoding of this problem in QAOA is quite troublesome, and, despite its exceptionally high approximability (having a fully polynomial approximation scheme), it is not known to belong to W-MaxSNP or to be L-reducible to any unconstrained problem. Finally, we aim to extend our testing to a wider variety of NP-hard problems and evaluate the framework's performance in environments with realistic hardware noise.

References

[1] J. Castaneda, E. Ghorbani, M. Ammouriova, J. Panadero, and A. A. Juan, "Optimizing transport logistics under uncertainty with simheuristics: Concepts, review and trends," *Logistics*, vol. 6, no. 3, 2022.

[2] I. Yevseyeva, E. B. Lenselink, A. de Vries, A. P. IJzerman, A. H. Deutz, and M. T. M. Emmerich, "Application of portfolio optimization to drug discovery," *Information Sciences*, vol. 475, pp. 29–43, 2019.

[3] C. H. Papadimitriou, *Computational complexity*. GBR: John Wiley and Sons Ltd., 2003, p. 260–265.

[4] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[5] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. Volume 2 - 2014, 2014.

[6] S. Hadfield, Z. Wang, E. G. Rieffel, B. O'Gorman, D. Venturelli, and R. Biswas, "Quantum Approximate Optimization with Hard and Soft Constraints," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017.

[7] D. Bucher, J. Stein, S. Feld, and C. Linnhoff-Popien, "Penalty-free approach to accelerating constrained quantum optimization," *Phys. Rev. A*, vol. 112, p. 062605, Dec 2025.

[8] B. Bakó, A. Glos, Ö. Salehi, and Z. Zimborás, "Prog-QAOA: Framework for resource-efficient quantum optimization through classical programs," *Quantum*, vol. 9, p. 1663, 2025.

[9] A. Kundu, L. Botelho, and A. Glos, "Hamiltonian-oriented homotopy quantum approximate optimization algorithm," *Phys. Rev. A*, vol. 109, p. 022611, 2024.

[10] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[11] C. H. Papadimitriou and M. Yannakakis, “Optimization, approximation, and complexity classes,” *Journal of Computer and System Sciences*, vol. 43, no. 3, pp. 425–440, 1991.
[12] A. B., H. {van Maaren}, T. Walsh, and H. Heule, Eds., *Handbook of Satisfiability*. IOS Press, Feb. 2009.
[13] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, “Proof verification and the hardness of approximation problems,” *J. ACM*, vol. 45, no. 3, p. 501–555, 1998.
[14] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. Faehrmann, B. Meynard-Piganeau, and J. Eisert, “Stochastic gradient descent for hybrid quantum-classical optimization,” *Quantum*, vol. 4, p. 314, 08 2020.
[15] E. Farhi, E. Gamarnik, and S. Gutmann, “The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples,” 2020.
[16] F. W. Glover and G. A. Kochenberger, “A tutorial on formulating qubo models,” *ArXiv*, vol. abs/1811.11538, 2018.