

QUANTUM COMPUTING WITH SEQUENTIAL QUANTUM PROCESSORS THROUGH BIAS TERMS

Field

The invention relates to quantum computing, and in particular to solving optimization problems using quantum computing.

Background

Classical computers and methods struggle with the combinatorial complexity of optimization problems, such as network modelling, resource allocation, and traveling sales-man problems, making them inefficient due to scaling issues and increased computational demands. This has led to interest in quantum computing and algorithms, but these too face challenges such as hardware limitations, scalability problems, and high error rates.

It is therefore an object of the invention to present a quantum solution and to overcome at least in part the disadvantages of the prior art.

Summary

The present invention overcomes at least some drawbacks of prior art solutions.

According to an aspect, the present invention relates to a method for solving combinatorial optimization problems.

According to an aspect, the present invention relates to a computer-implemented method for solving a combinatorial optimization problem the solution of which is encoded in at least one energy state of a Hamiltonian, preferably an Ising spin-glass Hamiltonian, the method being configured to be implemented by at least one computerized system comprising at least one classical processing unit, at least one electromagnetic pulse generation module, at least one measurement module and at least one quantum processing unit, the quantum processing unit comprising a plurality of qubits and a plurality of qubit gates, the method comprising the following steps:

- a) Selecting, using at least one classical processing unit, at least one combinatorial optimization problem, in particular related to a maximum independent set problem for example network modelling, resource allocation, traveling sales-man problem, max-k-SAT, etc.;

- b) Providing, using at least one communication module, an initial Hamiltonian comprising an initial ground state, said initial ground state being configured to be computationally initialized over at least one set of qubits from a plurality of qubits of a quantum processing unit, the initialization of the ground state being done using at least one state-preparation process, such as quantum annealing, adiabatic state preparation or counter-diabatic state preparation, the preparation process being configured to use at least one set of qubit gates from a plurality of qubit gates of the quantum processing unit to manipulate the quantum state of the set of qubits using a set of electromagnetic pulses, using an electromagnetic pulse generation module;
- c) Providing, using the communication module, a final Hamiltonian configured to encode the selected optimization problem, the final Hamiltonian comprising a set of energy states, preferably a set of low energy states, comprising a final energy state, preferably the lowest energy state of all the energy state of the final Hamiltonian, the final energy state being configured to encode a solution of the combinatorial optimization problem;
- d) Encoding, using the classical processing unit, the selected optimization problem to the final Hamiltonian;
- e) Iterating the following steps until the final energy state is reached based on a bias-field feedback mechanism:
- Initializing, using the quantum processing unit and the electromagnetic pulse generation module, the set of qubits in an initial configuration, the initial configuration comprising initial quantum states for each qubit of the set of qubits and initial interactions between each qubit of the set of qubits, the initial configuration of the set of qubits corresponding to the ground state of the initial Hamiltonian, this initialization step comprising at least:
 - i. Putting each qubit of the set of qubits into predetermined quantum states by emitting a plurality of electromagnetic pulses using the electromagnetic pulse generation module;
 - ii. Constructing a time dependent adiabatic Hamiltonian comprising a summation of the initial Hamiltonian and the final Hamiltonian with time dependent control parameters, the time dependent control parameters comprising a set of control electromagnetic

- pulses configured to drive the initial Hamiltonian to the final Hamiltonian using the electromagnetic pulse generation module;
- iii. Applying a series of operator transformations to the time dependent adiabatic Hamiltonian, the operator transformations comprising at least one recursive series of nested commutators, each commutator of the series of nested commutators involving at least one adiabatic Hamiltonian and its derivative with respect to the time dependent control parameters;
 - iv. Modifying the terms of the adiabatic Hamiltonian by incorporating contributions from the nested commutators to generate at least one counter-diabatic Hamiltonian;
 - v. Summing the adiabatic Hamiltonian with the counter-diabatic Hamiltonian to obtain a total Hamiltonian;
- Evolving the quantum state of the set of qubits according to the total Hamiltonian to reach at least one of the lowest energy states of the final Hamiltonian, the evolving step comprising:
 - i. Counter-diabatically evolving the set of qubits with the total Hamiltonian up to a predetermined final time which is equal to or less than the coherence time of the set of qubits, this evolving step comprising:
 - 1. Emitting a set of time-dependent electromagnetic pulses, using the electromagnetic pulse generation module, to drive the set of qubits under the total Hamiltonian, the set of time-dependent electromagnetic pulses being used to generate interactions among the qubits of the set of qubits according to the total Hamiltonian;
 - Measuring, using at least one measurement module, each qubit of the set of qubits to obtain at least one state probability distribution, the state probability distribution containing a set of pairs of unique bitstring and its associated probability, the unique bitstring being the measured states of the qubits in a computational basis, the probability being the number of times the unique bitstring was measured relative to the total number of total measurements;

- Post processing, using the classical processing unit, the result of the state probability distribution, the post processing comprising the following steps:
 - i. Building an energy probability distribution as a set of triplets, each triplet containing one unique bitstring, one energy value, which is the expectation value of the unique bitstring with respect to the final Hamiltonian, and the probability associated to the bitstring and retrieved from the state probability distribution;
 - ii. Generating a post-processed energy probability distribution by performing a mapping of the probabilities of each triplet over a predetermined scale;
- Calculating, using the classical processing unit, at least one expectation value from the post-processed energy probability distribution;
- Extracting, using the classical processing unit, a ground state solution from the calculated expectation value ;
- Constructing, using the classical processing unit, a bias-field Hamiltonian from the final ground state by performing a product of the calculated expectation values corresponding to each qubit and operator transformation, and by performing a summation of a plurality local bias-field Hamiltonians, each local bias-field Hamiltonian of the plurality of local bias-field Hamiltonians being obtained from at least one qubit of the set of qubits;
- Calculating, using the classical processing unit, the energy level of the ground state solution;
- Saving, in at least one storage module, the calculated energy level to a list comprising each iteration of ground state solution associated with its calculated energy level;
- Comparing, using the classical processing unit, the calculated energy level with the other calculated energy levels from the list, if the calculated energy level is equal to the lowest energy level already saved within the list, stopping the iterating step;

- Updating, using the classical processing unit, the initial Hamiltonian by summing the initial Hamiltonian with the constructed bias-field Hamiltonian;
- f) Obtaining, using the classical processing unit, the calculated expectation value from the last ground state solution;
- g) Determining, using the classical processing unit, the solution to the combinatorial optimization problem as being the identified expectation value.

According to another aspect, the present invention relates to a computer product program for solving a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian which, when executed by at least one computerized system according to the present invention, executes the method according to the present invention.

According to another aspect, the present invention relates to a non-volatile memory comprising at least one computer program product according to the present invention.

According to another aspect, the present invention relates to a computerized system configured to solve a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian, the system comprising:

- at least one classical processing unit configured to:
 - select at least one combinatorial optimization problem;
 - encode the selected optimization problem into a final Hamiltonian;
 - post-process state probability distributions and generate energy probability distributions;
 - calculate expectation values from energy probability distributions;
 - extract ground state solutions and construct bias-field Hamiltonians;
 - compare calculated energy levels and update the initial Hamiltonian; and
 - determine the solution to the combinatorial optimization problem;
- at least one electromagnetic pulse generation module configured to:

- emit a plurality of electromagnetic pulses to initialize qubits into predetermined quantum states;
 - drive the set of qubits under the total Hamiltonian using time-dependent electromagnetic pulses;
- at least one measurement module configured to measure the quantum state of each qubit of a set of qubits to obtain state probability distributions;
- a quantum processing unit comprising:
 - a plurality of qubits; and
 - a plurality of qubit gates, preferably including single or multi-qubit gates,

wherein the quantum processing unit is configured to:

- initialize the set of qubits in an initial configuration corresponding to the ground state of the initial Hamiltonian;
 - evolve the initial Hamiltonian according to the total Hamiltonian to reach at least one low energy state of the final Hamiltonian;
 - and
 - provide measured states of the qubits in a computational basis;
- at least one communication module configured to provide the initial Hamiltonian, the final Hamiltonian;
- at least one storage module configured to store each iteration's ground state solution associated with its calculated energy level.

The present invention relates generally to quantum computing and specifically to a method for optimizing combinatorial problems by iterative bias (anti-bias)-field application using both analog and digital quantum computation methods. In addition, the invention combines the techniques of digitized-counter-diabatic protocols inspired from quantum control with the bias field concept to obtain optimal/sub-optimal or approximate solutions for combinatorial optimization problems. Certain embodiments are defined by the appended independent and dependent claims. Certain embodiments are defined by the appended independent and dependent claims.

The invention pertains to a computer-implemented method designed to enhance solutions for complex computational problems, such as combinatorial optimization

problems, quantum sampling, and other problems addressed by quantum computers, by utilizing hybrid analog and digital quantum computing techniques. In specific embodiments, this method integrates the solutions from analog quantum (or classical) devices with digital quantum computers, or vice versa, through bias terms. Here, the bias terms encode some information about the solution of the problem obtained from one hardware and serve as an input for the subsequent hardware. The subsequent hardware may either be the same device used iteratively or a different device. This integration exploits the individual capabilities of different hardware and enables the enhancement of solutions through successive iterations, utilizing the bias (anti-bias)-field derived from previous solutions to guide the computation towards better solution.

Before providing below a detailed review of embodiments of the technology, some optional characteristics that may be used in association or alternatively will be listed hereinafter:

According to an example, the present invention comprises, before the iterating step, a step of summing the initial Hamiltonian with at least one generic bias-field Hamiltonian, preferably the generic bias-field Hamiltonian being obtained by extracting a non-optimal solution of the optimization problem obtained from a quantum or a classical algorithm running on a quantum or classical hardware.

According to an example, the post-processing further comprises, before the step of generating a post-processed energy probability distribution, at least the following steps:

- Sorting the energy probability distribution, preferably in increasing energy value;
- Removing at least one portion of the energy probability distribution, said removed portion corresponding to triplets having an energy value higher than the remaining portion of the energy probability distribution to reduce the energy probability distribution, the remaining energy probability distribution comprising at least one triplet having the lowest energy value over the whole energy probability distribution;
- On the reduced energy probability distribution, performing at least one local search on each unique bitstring, where the local search comprises qubit gate operations on the qubits of the set of qubits to find a new unique bitstring having a lower energy than the unique bitstring with the lowest energy within the

reduced energy probability distribution, if the new unique bitstring is found, the triplet corresponding to the unique bitstring with the lowest energy is replaced in the energy probability distribution.

According to an example, the remaining energy bitstrings have larger probabilities.

According to an example, the electromagnetic pulses are global or local electromagnetic pulses regarding the set of qubits.

According to an example, the constructed bias-field Hamiltonian is an anti-bias-field Hamiltonian.

According to an example, the initial Hamiltonian is selected or generated by a user, or automatically generated using a simple initial Hamiltonian.

According to an example, the present invention further comprises, before the iterating step (105), a step of summing the initial Hamiltonian with at least one generic bias-field Hamiltonian obtained by extracting a non-optimal solution of the optimization problem from a quantum or a classical algorithm running on a quantum or classical hardware.

According to an example, the state-preparation process is selected from a group comprising: quantum annealing, adiabatic state preparation, and counter-diabatic state preparation.

According to an example, the constructed bias-field Hamiltonian is derived from solutions obtained using classical heuristic algorithms, quantum annealers, or for example iterative digitized-counter-diabatic quantum optimization (DCQO).

According to an example, the counter-diabatic Hamiltonian is based on a biased or anti-biased digitized-counter-diabatic protocol.

According to an example, the ground state solution is obtained using one or more techniques selected from a group comprising: adiabatic quantum optimization, quantum annealing, simulated annealing, simulated quantum annealing, and classical Ising Machine operations.

Description of the drawings

For a better understanding of the present technology, as well as other aspects and further features thereof, reference is made to the following description which is to be used in conjunction with the accompanying drawings, where:

Figure 1: A schematic diagram showing quantum computing with sequential quantum processors consisting of analog quantum annealer followed by a digital quantum computer solving a combinatorial optimization problem. The ordering of the quantum processors can be changed depending on the need and the problem to be solved.

Figure 2: A flowchart describing the quantum computing with sequential quantum processor through bias field for solving combinatorial optimization problems. The ordering of the quantum processors can be arbitrary. In the figure analog quantum processor is used as an initializer for digital quantum processor implementing the digitized-counterdiabatic quantum optimization (DCQO) protocol with bias field.

Figure 3: bf-DCQO for Ising spin-glass problem with long-range interaction. In (a) the schematic diagram of bf-DCQO is shown. In (b) the ground state success probability is plotted for system sizes between 10 and 20 qubits. For each system size, there are 400 randomly generated all-to-all connected spin-glass problems, which are taken from a normal distribution with mean 0 and variance 1. We show the scaling of both, bf-DCQO algorithm with 10 iterations and standard DCQO, with simulation parameters $\Delta t = 0.1$, $n_{\text{trot}} = 3$. In (c) the classification of the 400 instances using enhancement in success probability as criterion, where we count the number of instances successfully tackled by either bias or anti-bias field DCQO, and those that failed in both cases. In (d) emulation results for a randomly generated spin-glass as in previous cases. We performed 39 iterations of bf-DCQO, whose increasing approximation ratio is shown in the y-axis. For each iteration, we used $n_{\text{trot}} = 2$, $\theta_{\text{cutoff}} = 0.05$, $n_{\text{shots}} = 1000$ and IonQ forte noise model, accessed through IonQ-cloud. Furthermore, the associated all-to-all connected graph is shown.

Figure 4: Comparison between both simulated bf-DCQO and QAOA($p=3$). The analysis is done over 10 different random spin-glass and all-to-all connected instances, with varying system size between 10 and 20 qubits. We used 20 different random

initializations for QAOA and COBYLA optimizer with maximum number of iterations 300. It was contrasted to 10 iterations of bf-DCQO with $n_{\text{trot}} = 3$.

Figure 5: Experimental results. In (a) the ninth iteration of bf-DCQO for a randomly generated 36-node weighted MIS instance. For the simulations we used $n_{\text{trot}} = 3$, $\theta_{\text{cutoff}} = 0.012$, $n_{\text{shots}} = 1000$, whereas for the experimental case on IonQ Forte, we ran only the ninth iteration with $n_{\text{shots}} = 2500$ using debias error mitigation. The MIS size was 16 and we obtained an independent set of size 11, which can be seen in the drawn graph. In (b) the eighth iteration of bf-DCQO for a nearest-neighbor randomly generated 100 qubit spin-glass instance. For the simulations we used $n_{\text{trot}} = 2$, $\theta_{\text{cutoff}} = 0.05$ and $n_{\text{shots}} = 25000$. For the experimental results on IBM Brisbane, we used $n_{\text{shots}} = 1000$. Additionally, the circuit layout on the hardware is shown.

Figure 6 illustrates a flowchart of steps of a method according to an embodiment of the present invention.

Figure 7 illustrates schematically a system according to an embodiment of the present invention.

Short Description of the Invention

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This summary is not intended to identify key features of the invention, nor is it intended to be used to limit the scope of the invention.

The object of the invention was to provide a novel method for optimizing combinatorial problems.

This object is achieved by the features of the independent claims in the appended claims. Further embodiments and examples are apparent from the dependent claims, the detailed description, and the accompanying drawings of the figures.

The invention utilizes both analog quantum and digital quantum computers. Additionally, the invention can be employed to enhance the solutions derived from existing classical heuristic algorithms, including but not limited to simulated annealing

(SA), simulated quantum annealing (SQA), tabu search, and local search. In addition, the invention can also be used to improve the solutions obtained from classical Ising machines. In addition, the invention may include a digital quantum computing technique called biased (anti-biased) digitized-counterdiabatic protocols, which is an iterative scheme where a bias (anti-bias)-field derived from previous solutions is utilized to guide the optimization process toward a more accurate ground state determination.

In a first aspect, the invention refers to a computer-implemented method for solving a combinatorial optimization problem encoded in the ground state of an Ising spin-glass model.

In certain embodiments of the invention, the method comprises the steps of:

- a) providing or preparing an initial Hamiltonian, whose ground state can be easily prepared. In some embodiments, the selection of the initial Hamiltonian is done by the user. However, one can also automate this by choosing some simple initial Hamiltonian whose ground state can be easily prepared. In certain embodiments, the initial Hamiltonian is provided by a user performing the method. In other embodiments, the provision of the initial Hamiltonian is automated. In particular, a simple initial Hamiltonian may be chosen whose ground state can be easily prepared.
- b) Solving (evolving) the initial Hamiltonian for the ground state of the final Hamiltonian, in particular using a state-preparation method such as quantum annealing, adiabatic state preparation or counterdiabatic state preparation;
- c) extracting a ground state solution from the final Hamiltonian, in particular via sampling;
- d) constructing a bias(anti-bias)-field from the solution;
- e) modifying the initial Hamiltonian with the bias (anti-bias)-field;
- f) performing an iterative digitized-counterdiabatic quantum optimization (DCQO) on a programmable quantum computer; and
- g) determining the solution to the combinatorial optimization problem based on the best outcome of the iterations.

In certain embodiments of the invention, the ground state solution of step c) above is obtained using one or more techniques selected from the group consisting of: adiabatic quantum optimization, quantum annealing, simulated annealing, simulated quantum annealing, and classical Ising Machine operations.

In certain embodiments of the invention, the programmable quantum computer utilizes one or both of standard one-qubit gates and two-qubit gates and digital-analog quantum computing techniques to perform DCQO. A standard quantum gate can be any arbitrary single qubit rotations like Rx, Ry, Rz and any static two-qubit entangling gates like CNOT, CZ and any two-qubit dynamic gates like CPHASE, cross-resonance (CR) gate or Molmer-Sorensen gate (MS).

In certain embodiments of the invention, the bias(anti-bias)-field is iteratively refined with each DCQO cycle to progressively optimize towards the ground state.

In a second aspect, the invention refers to a system for implementing the method described herein, comprising: a computational module for encoding a combinatorial optimization problem; a quantum processing unit capable of executing various annealing techniques; a sampling module for solution extraction; a bias(anti-bias)-field construction unit; and a programmable quantum computer designed to perform DCQO and iterative bias(anti-bias)-field refinement.

In a third aspect, the invention refers to a data processing apparatus/device/system comprising means for carrying out [the steps of] the method described herein.

In a fourth aspect, the invention refers to a computer program [product] comprising instructions which, when the program is executed by a computer, cause the computer to carry out [the steps of] the method described herein.

In a fifth aspect, the invention refers to a computer-readable [storage] medium comprising instructions which, when executed by a computer, cause the computer to carry out [the steps of] the method described herein.

Detailed Description of the Invention

Combinatorial optimization problems are pervasive in scientific and engineering domains. Traditional methods such as quantum annealing have been used to solve such problems encoded in the ground state of an Ising spin-glass model. However,

these methods often struggle with local minima and require extensive computational resources. The present invention addresses these challenges by iteratively refining the Hamiltonian with information from prior “solutions”, improving the likelihood of ground state convergence.

In particular, the invention introduces a method called bias field digitized counterdiabatic quantum optimization (bf-DCQO) for solving combinatorial optimization problems on digital quantum computers. This method integrates auxiliary counterdiabatic (CD) terms into the adiabatic Hamiltonian and introduces bias terms derived from classical methods, quantum annealers, or iterative DCQO applications. This approach addresses the limitations of current quantum computers, particularly their limited coherence times, and improves the time to solution for large-scale problems compared to traditional DCQO and finite-time adiabatic quantum optimization.

The bf-DCQO method is entirely quantum, avoiding the trainability issues of variational quantum optimization algorithms. It significantly outperforms the Quantum Approximate Optimization Algorithm (QAOA) in success probability and approximation ratio. Experimental implementation on a trapped-ion quantum computer and a superconducting processor demonstrated its effectiveness on a maximum weighted independent set problem with 36 qubits and a spin-glass problem on a heavy-hex lattice with 100 qubits. This represents the largest such problem solved on a gate-based quantum computer using a purely quantum algorithm.

In one aspect, the invention refers to a computer-implemented method for solving a combinatorial optimization problem, in particular encoded in the ground state of an Ising spin-glass model. In certain embodiments, the method comprises the steps of:

- a) providing an initial Hamiltonian;
- b) evolving or solving the initial Hamiltonian for the ground state to obtain a final Hamiltonian; Solving the initial Hamiltonian for the ground state of the final Hamiltonian, in particular using a state-preparation methods such as quantum annealing, adiabatic state preparation or counterdiabatic state preparation;
- c) extracting a ground state solution from the final Hamiltonian, in particular using sampling;

- d) constructing a bias(anti-bias)-field from the ground state solution;
- e) modifying the initial Hamiltonian with a bias(anti-bias)-field;
- f) performing an iterative digitized-counterdiabatic quantum optimization (DCQO) on a programmable quantum computer; and
- g) determining the solution to the combinatorial optimization problem based on the best outcome of the iterations.

A particular embodiment of the method of the invention (algorithm) is summarized in the steps given below.

1: Problem Encoding: A combinatorial optimization problem is encoded in the ground state of an Ising spin-glass model.

2: Analog component of the algorithm: Utilizing hardware capable of solving a ground state of an Ising spin-glass model. This can be digital classical computers with simulated annealing, simulated quantum annealing, or analog classical Ising machines or analog quantum processors.

In the next step, the analog quantum processor component is primarily described. Ways of extracting solutions from a classical computer or simulator, which will subsequently be used as input for the digital quantum computing segment, are known in the state of the art.

2 (a): Input for analog quantum annealers or adiabatic quantum computers: To solve for the ground state, we utilize adiabatic quantum optimization technique. This consist of initial Hamiltonian H_i , whose ground state is easily prepared, a problem Hamiltonian H_p (as in step-1 encoding the solution to the combinatorial problem of interest, and a time-dependent scheduling function $\lambda(t)$ defining the adiabatic path.

$$H(\lambda) = (1 - \lambda)H_i + \lambda(t) H_p \quad (1)$$

Here, the system evolves from the ground state of initial Hamiltonian towards the ground state of the problem Hamiltonian. The annealing time T decides the total evolution time or the run time of this algorithm.

2 (b): Solution Extraction from the analog quantum annealers or adiabatic quantum computers: Following the evolution period T , the solution, represented as a probability distribution measured on the computational basis, is extracted from the hardware using sampling techniques.

3: Bias(anti-bias)-Field Construction: The extracted solution is used to construct a bias(anti-bias)-field that encodes the solution information. This bias(anti-bias)-field will be used in the following step by modifying the initial Hamiltonian such that it provides a good initial starting point for the digitized-counterdiabatic protocol in the next step.

4: Digital component of the algorithm: Utilizing programmable quantum computers based on hardware technologies including superconducting, trapped-ions, neutral atoms, photonics and spin qubits.

4 (a): Hamiltonian: A programmable quantum computer solves the same problem Hamiltonian, modified by the sum of the initial Hamiltonian and the bias(anti-bias)-field Hamiltonian, reflecting the solution from the previous iteration. In addition, we introduce counterdiabatic terms to speed-up the evolution. The total Hamiltonian takes the form

$$H(t) = [1 - \lambda(t)](H_i + H_{bias}) + \lambda(t)H_p + \dot{\lambda}A_\lambda \quad (2)$$

Here, $H_{bias} = \sum \tilde{h}_i^z \sigma_i^z$, and the bias-field coefficients are obtained from the previous solution as $\tilde{h}_i^z = \langle \sigma_i^z \rangle$. And for the antibias case $H_{anti-bias} = -\sum \tilde{h}_i^z \sigma_i^z$. The added counterdiabatic Hamiltonian is $H_{CD} = \dot{\lambda}A_\lambda$, and A_λ is the approximate adiabatic gauge potential obtained variationally. For a large evolution time the counterdiabatic term can be neglected.

5. Iterative Optimization: The programmable quantum computer performs digitized-counterdiabatic quantum optimization (DCQO) using standard one and two-qubit gates or digital-analog quantum computing techniques. This process is repeated iteratively, with the bias(anti-bias)-field as in Eq. 2 updated at each iteration, until certain criteria such as the number of iterations are met.

6. Final Solution Determination: After completing the iterative DCQO cycles with the bias(anti-bias)-field, the method/algorithm extracts the best solution among all iterations. The best solution can be the state with minimum energy, or highest

approximation ratio, or highest ground state success probability or state with highest probability.

Particular embodiments of the invention include a quantum computing environment capable of both analog and digital operations, with a dedicated module for bias(anti-bias)-field calculation and integration. A user interface provides controls for setting the initial parameters of the combinatorial optimization problem, and monitoring tools allow for real-time observation of the iterative optimization process.

The invention, a bias(anti-bias)-field iterative process, provides significant advantages over existing optimization techniques, both quantum and classical optimization techniques, including enhanced convergence rates, reduced susceptibility to local minima, and improved computational efficiency.

Quantum computing is a field of computation, which aims at outperforming classical computation by exploiting quantum mechanical phenomena. In this regard, it is necessary to introduce a qubit, which is a basic unit of quantum information.

The qubit may be considered as the quantum analog of a classical bit. It is representative of a physical system that may be in two different states, generally denoted by $|0\rangle$ and $|1\rangle$, as well as in a superposition of those two states, e.g., $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

This plays an important role in the development of quantum algorithms outperforming classical algorithms. One example of the physical device that may be used as a qubit is an electron spin.

Unitary operation: The time evolution of qubits is specified by a unitary operator acting on qubit states. The unitary operator plays the role of a gate in the quantum computing. In general, these gates are generated via some Hamiltonian that makes a qubit system to evolve in time.

A quantum circuit is a model for quantum computation in which a computation is a sequence of quantum gates, which are reversible transformations on a quantum mechanical analog of an n-bit register. This register consists of n qubits.

Circuit depth is defined as the number of parallel unitary operations that can be performed for a given algorithm to run on a given quantum hardware.

Coherence time is the duration over which a quantum system, such as a qubit in a quantum computer, maintains its quantum state without significant decoherence. Decoherence is the process by which a quantum system loses its quantum mechanical properties, typically due to interactions with its external environment, leading to the loss of superposition and entanglement. Coherence time sets a limit on the time available to perform quantum operations or computations before the quantum information is degraded.

Gate error refers to inaccuracies that occur during the execution of quantum gates, which are fundamental operations on qubits. These errors can be caused by a variety of factors including environmental disturbances, imperfections in qubit fabrication, or inaccuracies in the control mechanisms that execute quantum operations. Gate errors are significant because they directly impact the fidelity of quantum operations, thereby affecting the accuracy and reliability of quantum computations. As such, mitigating gate errors is crucial, involving strategies like improving qubit design to enhance stability, refining control techniques to reduce execution imperfections, and implementing quantum error correction protocols to detect and correct errors without disturbing the quantum state.

Gate fidelity refers to the accuracy with which quantum gates, the basic operations that manipulate qubits—perform their intended tasks. It quantifies the degree of closeness between the theoretical or ideal quantum gate operation and the actual operation as implemented in a quantum computing system. High fidelity is crucial for practical quantum computations as it ensures the reliability of quantum algorithms, particularly in systems that require a complex sequence of operations.

State fidelity refers to a measure of the closeness between two quantum states. It is often used to quantify the accuracy of a quantum operation or to compare the theoretical and experimental states of a quantum system. More specifically, the state fidelity between two quantum states, represented as ρ and σ , is defined by the formula:

$$F(\rho, \sigma) = \left(\text{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2$$

Hamiltonian and k-local terms: A Hamiltonian is an operator representing the total energy of a quantum system, essential for describing the system's evolution over time. Particularly relevant are k-local Hamiltonians, where each term in the Hamiltonian involves interactions among at most k qubits. This concept is crucial in the realm of quantum simulations and algorithms, as it realistically models physical systems which typically exhibit local interactions. For instance, a 2-local Hamiltonian includes terms that describe interactions between pairs of qubits but not three or more simultaneously.

A quantum gate is a fixed unitary evolution. An example is a multi-qubit operation:

$$U_N = e^{i\pi/4} \sigma_1^z \sigma_2^z \quad (1).$$

An analog block (or gate) is a parametrized entangling unitary evolution with more than one parameter. An example is a two-parameter-dependent multiqubit operation in trapped ions (global MS gate) as

$$U_{MS}^n(\theta, \phi) = e^{-i\frac{\theta}{4}(\cos\phi S_x + \sin\phi S_y)^2} \quad (2).$$

Where $S_{x(y)} = \sum_i \sigma_i^{x(y)}$

A digital block (or gate) is a fixed unitary evolution up to a set of local rotations upto 2-qubits. The examples are parameter-fixed entangling quantum gates and single qubit rotations with arbitrary angles as

$$U_N(\phi) = (e^{i\phi_1} \sigma_1^z \otimes e^{i\phi_2} \sigma_2^z) e^{i\pi/4} \sigma_1^z \sigma_2^z, \text{ and} \quad (3).$$

$$U_i(\phi) = e^{i\phi} \sigma_1^z.$$

$$U_{MS}^2(\theta, 0) = e^{-i\theta/4} \sigma_1^x \sigma_2^x.$$

The problem Hamiltonian (also referred to here as target Hamiltonian) is the Hamiltonian that needs to be mapped to the quantum computer or simulator and simulated using DACQO algorithm.

The term Initial Hamiltonian refers to the system Hamiltonian that is designed to have a simple and known ground state. It serves as the starting point of the annealing process. As the annealing progresses, the Hamiltonian of the system gradually transforms from this initial Hamiltonian to a problem Hamiltonian.

The term Final Hamiltonian refers to the second sub-Hamiltonian that encodes the solution to the problem being solved in its ground state. In quantum annealing, the system evolves towards this Hamiltonian such that the quantum state will end up in the ground state of this Hamiltonian, effectively solving the problem.

The term Auxiliary Hamiltonian refers to the third sub-Hamiltonian that consists of local fields over each qubit provided. While its specific purpose might vary depending on the problem and the annealing protocol, auxiliary Hamiltonians are typically introduced to assist the annealing process and ensure a smooth transition from the initial to the problem Hamiltonian.

The term Instantaneous Hamiltonian refers to the Hamiltonian at any given point in time during the annealing or evolution process. As the annealing progresses, the system Hamiltonian changes from the initial to the final Hamiltonian. The Hamiltonian at any particular moment during this transformation can be considered the instantaneous Hamiltonian.

Expectation value is the average of the measurable quantity of a quantum system which can include the energy, momentum or other dynamical quantity. It is obtained by the following relation

$$\frac{\langle \psi(t) | O(t) | \psi(t) \rangle}{|\langle \psi(t) | \psi(t) \rangle|^2}.$$

wherein $O(t)$ is the time dependent (or maybe independent) operator of interest and $\psi(t)$ is the time dependent wavefunction of the quantum system that determines its state at a particular time t .

A quantum processor is a programable quantum devices composed of several informational units (qubits) that can be tuned in order to perform quantum algorithms.

Trotterization is a technique used in quantum computing to simulate the evolution of quantum systems governed by Hamiltonians that are sums of non-commuting terms. It breaks down the exponential of a sum of operators into a product of exponentials of these operators, as $e^{(A+B)t} \approx \left(e^{\frac{At}{n}} e^{\frac{Bt}{n}} \right)^n$ allowing for an approximate simulation of the quantum system's dynamics. Here, A and B are, generally, non-commuting operators, t represents time, and n is the number of subdivisions or steps in the approximation. As n approaches infinity, this approximation becomes exact. This method is based on the Trotter-Suzuki formula, which provides a way to approximate the evolution operator of the entire system through a sequence of simpler operations that are easier to implement on a quantum computer.

Among the approaches to do quantum computing, there are mainly two, analog and digital methods of quantum computing. The analog method aims to design the hardware to mimic a certain problem (Hamiltonian). While being high in accuracy, it has the disadvantage that it is specific to a given problem and lacks flexibility. In digital quantum computation (DQC), the digital quantum gates (single and two-qubit) are used to tune the hardware to reach a certain target Hamiltonian.

Digital quantum computing (DQC) is based on the application of sequences of single-qubit and multi-qubit gates. These operations are termed “digital blocks” and the protocol is known as a digital quantum algorithm. DQC is limited by the fact that it is resource-consuming and does not display robustness to errors in the computation. This makes DQC challenging to implement important and complex problems and reach quantum advantage.

Analog quantum computing involves configuring quantum systems to emulate other quantum systems directly, making it highly effective for simulating quantum physics phenomena. Unlike digital quantum computing, which uses quantum bits (qubits) to perform calculations in a manner analogous to binary bits in classical computing, analog quantum computing operates by leveraging the natural quantum mechanical properties of the system. This method allows for a more intuitive representation of quantum simulations, particularly useful in fields such as material science and chemistry where the inherent complexities of quantum interactions are a central focus.

Digital-Analog Quantum Computing (DAQC) is an alternative approach, which uses single and two qubit gates as digital blocks and multi-qubit gates as analog blocks aiming to solve the problem with fewer resources (reduced circuit depth) than is required by DQC. Moreover, due to reduced circuit depth it incurs less error in the simulation compared to the DQC paradigm and is more accurate.

Quantum annealing is an analog computation method that may be used to find a low-energy state, typically preferably the ground state, of a system. Quantum annealing is used for problems where the search space is discrete (e.g. combinatorial optimization problems) with many local minima. Similar in concept to classical annealing, the method relies on the underlying principle that natural systems tend towards lower energy states because lower energy states are more stable. However, while classical annealing uses classical thermal fluctuations to guide a system to a low-energy state and ideally its global energy minimum, quantum annealing may use quantum effects, such as quantum tunneling, to reach a global energy minimum more accurately and/or more quickly than classical annealing. In quantum annealing thermal effects and other noise may be present to aid the annealing. However, the final low-energy state may not be the global energy minimum. Adiabatic quantum computation, therefore, may be considered a special case of quantum annealing for which the system, ideally, begins and remains in its ground state throughout an adiabatic evolution. Thus, those of skill in the art will appreciate that quantum annealing systems and methods may generally be implemented on an adiabatic quantum computer. Throughout this description, any reference to quantum annealing is related to adiabatic quantum computation unless the context requires otherwise.

Quantum annealing uses quantum mechanics as a source of disorder during the annealing process. The considered problem (i.e. the problem to be solved) is encoded in a Hamiltonian H_P , and the algorithm introduces quantum effects by adding a disordering Hamiltonian H_0 that does not commute with H_P .

A quantum annealer is a programable quantum device that is capable of interpolating an initial Hamiltonian H_0 at time $t=0$ with a final or problem Hamiltonian H_P at a time $t=T$. An appropriate design of a quantum annealer for solving the considered problem configured to produce a desired final Hamiltonian (problem Hamiltonian) comprises a

tunable scheduling function. The quantum annealer may comprise at least one scheduling function that is a time-dependent tunable parameter of the used hardware.

Quantum computers (quantum processors) may include quantum annealing processors, digitized quantum processors, gate-based processors, or adiabatic quantum computation.

Noisy intermediated scale (NISQ) devices: It refers to the current generation of quantum computers that are characterized by their relatively small number of qubits and the presence of noise and errors in their operations.

Optimization problems are central to many scientific, engineering, and economic applications and fall under a broad category of mathematical problems where the goal is to find the optimal solution from a set of available alternatives. The optimal solution is determined according to specific criteria, typically defined by an objective function that needs to be maximized or minimized. The computational complexity of finding optimal solutions in optimization problems are categorized into polynomial time (P), where the solution time scales polynomially with input size, or NP hard, that exhibit a solution time that can grow exponentially with the problem size, making them computationally challenging, especially for large-scale instances. Quantum computing aims to address this complexity.

A cost function defines the optimization problem. It is translated into a Hamiltonian, where the ground state represents the optimal solution. This cost function shapes the energy landscape. Quantum annealing aims to find the ground state, which corresponds to the cost function's minimum. Annealing starts with a simple "initial Hamiltonian." As the process progresses, the system is increasingly governed by the cost function's energy landscape. At the end of annealing, the quantum system's state gives a solution to the optimization problem, evaluated using the cost function.

Optimization algorithms may include simulated annealing, parallel tempering, Markov Chain Monte Carlo techniques, branch and bound algorithms, and greedy algorithms, which may be performed by a classical computer. Optimization algorithms may also include algorithms performed by a quantum computer, such as quantum annealing,

quantum approximate optimization algorithm (QAOA) or other noisy intermediate-scale quantum (NISQ) algorithms, quantum implemented fault-tolerant optimization methods, or other quantum optimization algorithms.

Quantum optimization algorithms are quantum algorithms that are used to solve optimization problems. A combinatorial optimization problem refers to the problem of finding the most efficient solution from a finite set of possibilities that satisfies certain criteria and constraints, often involving discrete or combinatorial structures like graphs, sequences, or configurations.

Variational quantum algorithms (VQA) are a family of hybrid quantum-classical algorithm, that mix the power of quantum computing with subroutines that contain classical optimization algorithms.

The variational quantum algorithms use the concept of the variational quantum ansatz. This is a parametrized quantum state, which can be produced by a parametrized quantum circuit in the case of digital quantum computers, or by a parametrized evolution in the case of analog quantum computers, or a combination of both in the case of digital-analog paradigm. Once the ansatz is built, then a cost or objective function is evaluated using the quantum processor. The objective function serves as the driving force of these algorithms, it encapsulates the problem to be solved and quantifies its outcome. Later, the expectation value of a cost function is the input for a classical optimization algorithm, which updates the parameters of the variational ansatz to obtain a quantum state that maximizes or minimizes the objective function according to the requirement of the problem.

Quadratic unconstrained binary optimization (QUBO) problems are a type of optimization problem and have a broad range of applications in various fields like machine learning, finance, and logistics. In a QUBO problem, one seeks to minimize a quadratic objective function of binary variables (variables that can take values of 0 or 1). The objective function is a sum of terms, each involving either a single variable or a product of two variables. These problems are unconstrained, meaning there are no explicit constraints on the variables apart from their binary nature.

Higher order unconstrained binary optimization (HUBO) problems involve optimizing polynomial functions of binary variables without explicit constraints. These problems are crucial in various fields, including operations research and physics. Solving HUBO problems requires advanced algorithms, with quantum computing offering promising new solutions. Quantum techniques exploit unique quantum properties to solve HUBO problems more efficiently than classical methods.

Ising spin glass Hamiltonian: The Ising spin glass model describes a system of spins that can interact with each other in a complex manner. In this model, spins are represented by variables that can take values of +1 or -1, typically denoting their 'up' or 'down' states. The model is defined on a lattice (or graph) where spins are placed on vertices and interactions between spins are represented by edges. These interactions can be ferromagnetic or antiferromagnetic, leading to complex energy landscapes. The energy of a spin configuration in the Ising model is given by a Hamiltonian, which is a function of the spin states and their interactions. The goal often involves finding the ground state of this Hamiltonian, which corresponds to the minimum energy configuration.

Problem encoding: Encoding a problem in quantum computing refers to the process of translating a specific computational problem or algorithm into a form that is suitable for a quantum computer to execute. For example, the QUBO problem can be encoded into the Ising spin glass Hamiltonian.

Adiabatic Quantum Computing (AQC) is a quantum computing methodology that leverages the adiabatic theorem of quantum mechanics. It starts with a quantum system in an easily identifiable ground state and then slowly evolves this system according to a changing Hamiltonian, a process known as adiabatic evolution. The final Hamiltonian of this evolution is designed such that its ground state encodes the solution to the computational problem of interest. AQC's strength lies in its robustness against certain types of errors and decoherence, as the system ideally remains in its lowest energy state throughout the computation. This approach is particularly effective for solving optimization problems and is closely related to quantum annealing. However, maintaining adiabaticity throughout the computation, especially for complex

problems, poses a significant challenge, as it often requires exceedingly slow evolution to prevent transitions to higher energy states.

Shortcuts to adiabaticity is a computational paradigm proposed to speed up Adiabatic evolution which can be infinitely slow for practical purposes using counterdiabatic driving. The concept behind counterdiabatic driving is to add an auxiliary Hamiltonian to the system such that the combined evolution is equivalent to the desired adiabatic process but achieved in a shorter time. This auxiliary Hamiltonian effectively counteracts the non-adiabatic transitions that would typically occur during a faster evolution.

Counterdiabatic quantum computing (CDQC) is a concept in quantum computing that aims to improve the efficiency and fidelity of quantum computations, particularly in the context of adiabatic quantum computing (AQC). Counterdiabatic (CD) driving, also known as transitionless quantum driving, is a technique used to speed up adiabatic processes without causing non-adiabatic transitions. It involves adding an auxiliary Hamiltonian (a counterdiabatic term) to the original Hamiltonian to cancel out the non-adiabatic transitions.

Adiabatic gauge potential is a concept in quantum mechanics that quantifies how the eigenstates of a system's Hamiltonian respond to changes in the Hamiltonian's parameters during an adiabatic process. It plays a critical role in ensuring that a quantum system remains in its ground state during adiabatic evolution, which is essential for applications like adiabatic quantum computing and quantum annealing. This potential is mathematically represented as $A(t) = i\langle\psi(t)|\frac{\partial}{\partial t}|\psi(t)\rangle$, where $|\psi(t)\rangle$ is an eigenstate of the Hamiltonian $H(t)$, and i is the imaginary unit. Understanding and managing this gauge potential is crucial for minimizing non-adiabatic transitions and optimizing the performance of quantum computational processes that rely on adiabatic changes.

In quantum mechanics and mathematical physics, a nested commutator is a higher-order expression that involves multiple layers of commutation between operators.

Commutators are a fundamental aspect of the algebraic structure of quantum mechanics and are defined for any two operators A and B as: $[A, B] = AB - BA$

A nested commutator involves commutators within commutators. For instance, a simple example of a nested commutator would be a second-order commutator like:

$$[[A, B], C] = (AB - BA)C - C(AB - BA) = ABC - BAC - CAB + CBA$$

Obtaining the exact adiabatic gauge potential ($A(t)$) for a many-body system is challenging. Its implementation is not optimal because, in many cases, $A(t)$ can contain exponentially many terms with nonlocal many-body interactions. An alternative approach is to consider the approximate form of the adiabatic gauge potential that can be obtained from the nested commutator method.

A scheduling function is defined as a function that controls the evolution of the Hamiltonian from an initial (in particular easily prepared) ground state (the initial Hamiltonian) to the final Hamiltonian that encodes the solution to the problem of interest over the course of the computation.

The time dependent scheduling function comprises at least one one-body CD term for adding an effective gauge potential that counters excitations arising from a non-adiabatic time evolution.

Trapped ion quantum computers: Trapped ions are a technology in the field of quantum computing, distinguished by their high-fidelity operations and long coherence times. This technology utilizes ions-atoms that have been ionized by adding or removing electrons confined and suspended in free space using electromagnetic fields. These trapped ions serve as qubits, the fundamental units of quantum information, in a quantum computer. Qubits are stored in stable electronic states of each ion, and quantum information can be transferred through the collective quantized motion of the ions in a shared trap. For single qubit operations, lasers are applied to induce coupling between the qubit states or, for entanglement between qubits, coupling between the internal qubit states and the external motional states.

Simulated Annealing (SA) is a probabilistic method used to find the approximate global optimum of a function. It mimics the physical process of heating a material and then slowly cooling it to minimize systemic energy.

Simulated Quantum Annealing (SQA) is a classical computational technique that mimics the quantum annealing process to find solutions to optimization problems, utilizing concepts of quantum mechanics like tunneling and entanglement.

Analog Classical Ising Machines are physical devices that solve optimization problems modeled as Ising spins using analog hardware. These machines map problems directly onto a network of interacting spins and use natural dynamics to find low-energy states of the system.

Analog Quantum Processors use genuine quantum mechanical effects to perform computations. These processors operate by manipulating continuous variables rather than digital bits, directly harnessing quantum dynamics and states to solve specific problems.

In the context of the present specification, “device” is any computer hardware that is capable of running software appropriate to the relevant task at hand. Thus, some (non-limiting) examples of devices include personal computers (desktops, laptops, netbooks, etc.), smartphones, and tablets, as well as network equipment such as routers, switches, and gateways. It should be noted that a device acting as a device in the present context is not precluded from acting as a server to other devices. The use of the expression “a device” does not preclude multiple devices being used in receiving/sending, carrying out or causing to be carried out any task or request, or the consequences of any task or request, or steps of any method described herein.

In the context of the present specification, a “database” is any structured collection of data, irrespective of its particular structure, the database management software, or the computer hardware on which the data is stored, implemented or otherwise rendered available for use. A database may reside on the same hardware as the process that stores or makes use of the information stored in the database or it may reside on separate hardware, such as a dedicated server or plurality of servers. It can be said that a database is a logically ordered collection of structured data kept electronically in a computer system.

In the context of the present specification, the expression “information” includes information of any nature or kind whatsoever capable of being stored in a database. Thus information includes, but is not limited to audiovisual works (images, movies, sound records, presentations etc.), data (location data, numerical data, etc.), text (opinions, comments, questions, messages, labels, etc.), documents, spreadsheets, lists of words, etc.

In the context of the present specification, the expression “component” or “module” is meant to include software (appropriate to a particular hardware context) that is both necessary and sufficient to achieve the specific function(s) being referenced.

In the context of the present specification, the expression “computer usable information storage medium” or “non-volatile memory” is intended to include media of any nature and kind whatsoever, including RAM, ROM, disks (CD-ROMs, DVDs, floppy disks, hard drivers, etc.), USB keys, solid state-drives, tape drives, etc.

In the context of the present specification, the words “first”, “second”, “third”, etc. have been used as adjectives only for the purpose of allowing for distinction between the nouns that they modify from one another, and not for the purpose of describing any particular relationship between those nouns. Thus, for example, it should be understood that, the use of the terms “first module” and “third module” is not intended to imply any particular order, type, chronology, hierarchy or ranking (for example) of/between the modules, nor is their use (by itself) intended imply that any “second module” must necessarily exist in any given situation. Further, as is discussed herein in other contexts, reference to a “first” element and a “second” element does not preclude the two elements from being the same actual real-world element. Thus, for example, in some instances, a “first” module and a “second” module may be the same software and/or hardware, in other cases they may be different software and/or hardware.

Implementations of the present technology each have at least one of the above-mentioned object and/or aspects, but do not necessarily have all of them. It should be understood that some aspects of the present technology that have resulted from

attempting to attain the above-mentioned object may not satisfy this object and/or may satisfy other objects not specifically recited herein.

Additional and/or alternative features, aspects and advantages of implementations of the present technology will become apparent from the following description, the accompanying drawings and the appended claims.

The examples and conditional language recited herein are principally intended to aid the reader in understanding the principles of the present technology and not to limit its scope to such specifically recited examples and conditions. It will be appreciated that those skilled in the art may devise various arrangements which, although not explicitly described or shown herein, nonetheless embody the principles of the present technology and are included within its spirit and scope.

Furthermore, as an aid to understanding, the following description may describe relatively simplified implementations of the present technology. As persons skilled in the art would understand, various implementations of the present technology may be of a greater complexity.

In some cases, what are believed to be helpful examples of modifications to the present technology may also be set forth. This is done merely as an aid to understanding, and, again, not to define the scope or set forth the bounds of the present technology. These modifications are not an exhaustive list, and a person skilled in the art may make other modifications while nonetheless remaining within the scope of the present technology. Further, where no examples of modifications have been set forth, it should not be interpreted that no modifications are possible and/or that what is described is the sole manner of implementing that element of the present technology.

Moreover, all statements herein reciting principles, aspects, and implementations of the present technology, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof, whether they are currently known or developed in the future. Thus, for example, it will be appreciated by those skilled in the art that any block diagrams herein represent conceptual views

of illustrative circuitry embodying the principles of the present technology. Similarly, it will be appreciated that any flowcharts, flow diagrams, state transition diagrams, pseudo-code, and the like represent various processes which may be substantially represented in computer-readable media and so executed by a computer or processor, whether or not such computer or processor is explicitly shown.

The functions of the various elements shown in the figures, including any functional block labeled as a "processor" or a "processing module" or a "processing unit", may be provided through the use of dedicated hardware as well as hardware capable of executing software in association with appropriate software. When provided by a processor, the functions may be provided by a single dedicated processor, by a single shared processor, or by a plurality of individual processors, some of which may be shared. In some embodiments of the present technology, the processor may be a general purpose processor, such as a central processing unit (CPU), also called a classical processing unit, or a processor dedicated to a specific purpose, such as a digital signal processor (DSP). Moreover, explicit use of the term a "processor" should not be construed to refer exclusively to hardware capable of executing software, and may implicitly include, without limitation, application specific integrated circuit (ASIC), field programmable gate array (FPGA), read-only memory (ROM) for storing software, random access memory (RAM), and non-volatile storage. Other hardware, conventional and/or custom, may also be included.

Software modules, or simply modules which are implied to be software, may be represented herein as any combination of flowchart elements or other elements indicating performance of process steps and/or textual description. Such modules may be executed by hardware that is expressly or implicitly shown. Moreover, it should be understood that module may include for example, but without being limitative, computer program logic, computer program instructions, software, stack, firmware, hardware circuitry or a combination thereof which provides the required capabilities. With these fundamentals in place, we will now consider some non-limiting examples to illustrate various implementations of aspects of the present technology.

According to an embodiment, the present invention relates to a computer-implemented method for solving a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian, preferably of an Ising spin-glass Hamiltonian.

Preferably, the method is configured to be implemented by a computerized system. This system, as described hereafter, may comprise at least one classical processing unit, at least one electromagnetic pulse generation module, at least one measurement module, and at least one quantum processing unit. Advantageously, and as described hereafter, the quantum processing unit comprises a plurality of qubits and a plurality of qubit gates. This plurality of qubit gates can comprise single and/or multi-qubit gates.

According to an embodiment, the method begins by selecting a combinatorial optimization problem, such as a maximum independent set problem, a network modeling problem, a resource allocation problem, a traveling salesman problem, or a max-k-SAT problem, etc. The selected optimization problem may be then encoded to provide an initial Hamiltonian comprising an initial ground state. This initial ground state can be configured to computationally initialize over at least one set of qubits from the plurality of qubits of the quantum processing unit using advantageously a state-preparation process such as quantum annealing, adiabatic state preparation, or counter-diabatic state preparation. The preparation process utilizes preferably at least one set of qubit gates from the plurality of qubit gates of the quantum processing unit to manipulate the quantum state of the set of qubits.

According to an embodiment, next, a final Hamiltonian is provided, configured to encode the selected optimization problem. The final Hamiltonian, also called the problem Hamiltonian, comprises a set of energy states, preferably a set of low energy states, including a final energy state, preferably the lowest energy state of all the energy states of the final Hamiltonian. This final energy state is configured to encode a solution to the combinatorial optimization problem.

Preferably, the selected optimization problem is then encoded, and the method proceeds by iterating a plurality of steps until the final energy state is reached based on a bias-field feedback mechanism. The bias-field feedback mechanism is advantageously used to update the initial Hamiltonian based on a ground state solution extracted in each iteration.

According to an embodiment, and as illustrated by figure 6, the present invention relates to a computer-implemented method 100 for solving a combinatorial optimization problem. Preferably, this problem is encoded in at least one energy state of an Hamiltonian. Advantageously, this problem is encoded in one of the lowest

energy state of this Hamiltonian. Preferably, this Hamiltonian is an Ising spin-glass Hamiltonian.

According to an embodiment, and as illustrated by figure 7, the method 100 is configured to be implemented by a computerized system 200.

According to an embodiment, the method 100 comprises at least the following steps:

- a) Selecting 101, using the classical processing unit 210, at least one combinatorial optimization problem, for example using a user interface;
- b) Providing 102, using the communication module 205, an initial Hamiltonian comprising an initial ground state, said initial ground state being configured to be computationally initialized over at least one set of qubits from a plurality of qubits of a quantum processing unit; Preferably the initialization of the ground state is done using at least one state-preparation process; advantageously, the preparation process is configured to use at least one set of qubit gates from the plurality of qubit gates to manipulate the quantum state of the set of qubits; For example, the quantum state of the qubits can be manipulated using a set of electromagnetic pulses, via the electromagnetic pulse generation module 202;
- c) Providing 103, using the communication module 205, a final Hamiltonian configured to encode the selected optimization problem; Preferably, the final Hamiltonian comprises a set of energy states comprising a final energy state being configured to encode a solution of the combinatorial optimization problem;
- d) Encoding 104, using the classical processing unit 201, the selected optimization problem to the final Hamiltonian ;
- e) Optionally, summing the initial Hamiltonian with at least one generic bias-field Hamiltonian; Preferably, the generic bias-field Hamiltonian can be obtained by extracting a non-optimal solution of the optimization problem obtained from a quantum or a classical algorithm running on a quantum or classical hardware;
- f) Iterating 105 the following steps until the final energy state is reached based on a bias-field feedback mechanism:
 - Initializing, using the quantum processing unit 204 and the electromagnetic pulse generation module 202, the set of qubits in an initial configuration; Preferably, the initial configuration comprises initial quantum states for each qubit of the set of qubits and initial interactions

between each qubit of the set of qubits; Advantageously, the initial configuration of the set of qubits corresponds to the ground state of the initial Hamiltonian; For example, this initialization step can comprise at least:

- i. Putting each qubit of the set of qubits into predetermined quantum states by emitting a plurality of electromagnetic pulses using the electromagnetic pulse generation module 202; Preferably, these electromagnetic pulses can be global or local electromagnetic pulses regarding the set of qubits;
 - ii. Constructing a time-dependent adiabatic Hamiltonian comprising a summation of the initial Hamiltonian and the final Hamiltonian with time-dependent control parameters; Preferably, the time-dependent control parameters comprises a set of control electromagnetic pulses configured to drive the initial Hamiltonian to the final Hamiltonian using the electromagnetic pulse generation module 202;
 - iii. Applying a series of operator transformations to the time-dependent adiabatic Hamiltonian; Preferably, the operator transformations comprises at least one recursive series of nested commutators; Advantageously, each commutator of the series of nested commutators involves at least one adiabatic Hamiltonian and its derivative with respect to the time-dependent control parameters;
 - iv. Modifying the terms of the adiabatic Hamiltonian by incorporating contributions from the nested commutators to generate at least one counter-diabatic Hamiltonian;
 - v. Summing the adiabatic Hamiltonian with the counter-diabatic Hamiltonian to obtain a total Hamiltonian;
- Evolving the quantum state of the set of qubits according to the total Hamiltonian to reach at least one of the lowest energy states of the final Hamiltonian, the evolving step comprising:
 - i. Counter-diabatically evolving the set of qubits with the total Hamiltonian up to a predetermined final time which is equal to or

less than the coherence time of the set of qubits; Preferably, this evolving step comprises:

1. Emitting a set of time-dependent electromagnetic pulses, using the electromagnetic pulse generation module 202, to drive the set of qubits under the total Hamiltonian; Advantageously, the set of time-dependent electromagnetic pulses are configured to generate interactions among the qubits of the set of qubits according to the total Hamiltonian;
- Measuring, using the measurement module 203, each qubit, preferably each qubit quantum state, of the set of qubits to obtain at least one state probability distribution: Advantageously, the state probability distribution comprises a set of pairs of unique bitstring and its associated probability; Preferably, the unique bitstring is the measured states of the qubits in a computational basis; Advantageously, the probability is the number of times a unique bitstring was measured relative to the total number of total measurements;
 - Post-processing, using the classical processing unit 201, the result of the state probability distribution; Preferably, the post-processing comprises the following steps:
 - i. Building an energy probability distribution as a set of triplets; Preferably, each triplet comprises one unique bitstring, one energy value, which is the expectation value of the unique bitstring with respect to the final Hamiltonian, and the probability associated to the bitstring and retrieved from the state probability distribution;
 - ii. Optionally, sorting the energy probability distribution, preferably in increasing energy value;
 - iii. Optionally, removing at least one portion of the energy probability distribution; Preferably, said removed portion corresponds to triplets having an energy value higher than the remaining portion of the energy probability distribution to reduce the energy probability distribution; Advantageously, the remaining energy

- probability distribution comprises at least one triplet having the lowest energy value over the whole energy probability distribution;
- iv. Optionally, on the reduced energy probability distribution, performing at least one local search on each unique bitstring; Preferably, the local search comprises qubit gate operations on the qubits of the set of qubits to find a new unique bitstring having a lower energy than the unique bitstring with the lowest energy within the reduced energy probability distribution; Advantageously, if the new unique bitstring is found, the triplet corresponding to the unique bitstring with the lowest energy is replaced in the energy probability distribution;
- v. Generating a post-processed energy probability distribution by performing a mapping of the probabilities of each triplet over a predetermined scale, Preferably, the remaining energy bitstrings have larger probabilities;
- Calculating, using the classical processing unit 201, at least one expectation value from the post-processed energy probability distribution;
 - Extracting, using the classical processing unit 201, a ground state solution from the calculated expectation value ;
 - Constructing, using the classical processing unit 201, a bias-field Hamiltonian from the final ground state by performing a product of the calculated expectation values corresponding to each qubit and operator transformation, and by performing a summation of a plurality local bias-field Hamiltonians, preferably each local bias-field Hamiltonian of the plurality of local bias-field Hamiltonians is obtained at least one qubit of the set of qubits;
 - Calculating, using the classical processing unit 201, the energy level of the ground state solution;
 - Saving, in at least one storage module 206, the calculated energy level to a list comprising each iteration of ground state solution associated with its calculated energy level;

- Comparing, using the classical processing unit 201, the calculated energy level with the other calculated energy levels from the list, if the calculated energy level is equal to the lowest energy level already saved within the list, stopping the iterating step;
 - Updating, using the classical processing unit 201, the initial Hamiltonian by summing the initial Hamiltonian with the constructed bias-field Hamiltonian;
- g) Obtaining 106, using the classical processing unit 201, the calculated expectation value from the last ground state solution ;
- h) Determining 107, using the classical processing unit 201, the solution to the combinatorial optimization problem as being the identified expectation value.

The present invention offers several technical advantages. The method is scalable to large problem sizes due to the ability to leverage quantum computing techniques, which can handle a vast number of variables and potential solutions simultaneously. By encoding the optimization problem into an Ising spin-glass Hamiltonian, for example, the solution process can be significantly faster compared to classical algorithms for solving similar problems. This speedup is particularly noticeable when dealing with large and complex optimization problems. The approach can be applied to a wide range of combinatorial optimization problems, making it a versatile tool for solving various practical problems in fields such as logistics, finance, and computer science.

According to an embodiment, the present invention further comprises selecting or generating an initial Hamiltonian by a user or automatically generating it using a simple initial Hamiltonian whose ground state can be easily prepared regarding the computational capabilities of the quantum processing unit. The initial Hamiltonian can either be chosen by the user or generated automatically as part of the method. This initial Hamiltonian can be configured to serve as a starting point for the optimization process. The initial Hamiltonian can be defined as simple in nature, making it easier to prepare its ground state regarding the considered quantum hardware. When the initial Hamiltonian is automatically generated, it can be done using a simple initial Hamiltonian whose ground state can be easily prepared regarding the considered

quantum hardware. Preferably, this approach ensures that the optimization process starts with a manageable state, thereby improving the efficiency of the method.

According to an embodiment, and to summarize, the present invention is configured to encode the combinatorial optimization problem the low energy state of an Hamiltonian.

According to an embodiment, the choice of the state-preparation process can influence the efficiency and effectiveness of the method. For instance, quantum annealing gradually changes the Hamiltonian from an initial one that favors the high-energy states to a final one that favors the low-energy states. Adiabatic state preparation follows a similar principle but with a more abrupt change in the Hamiltonian. Counter-diabatic state preparation uses additional control fields to counteract the diabatic effects and maintain the set of qubits in an adiabatic evolution.

It is important to note that while the present application focuses on solving combinatorial optimization problems encoded in Ising spin-glass Hamiltonians, for example, the method can be extended to other types of optimization problems and Hamiltonian systems. Furthermore, the choice of the state-preparation process can be optimized based on the specific characteristics of the problem at hand, such as the size of the problem, the nature of the interactions between variables, and the desired accuracy of the solution.

According to an embodiment, the use of a biased or anti-biased digitized-counter-diabatic protocol offers several technical advantages. It allows for a more efficient exploration of the solution space, as the Hamiltonian guides the set of qubits towards promising regions. It reduces the computational complexity by focusing on relevant qubits configurations. It improves the convergence rate, as the system is less likely to get stuck in local minima. Preferably, the digitized-counter-diabatic protocol is applied adaptively, meaning that the bias or anti-bias is adjusted dynamically during the optimization process based on the current state of the set of qubits. This further enhances the efficiency and effectiveness of the method.

According to an embodiment, adiabatic quantum optimization is used as one of the techniques for obtaining the ground state solution. This technique involves gradually changing the parameters of the set of qubits from an initial state to a final state while keeping the system in an adiabatic regime, which allows for efficient optimization.

Preferably, quantum annealing is another technique that can be used in the method. Quantum annealing is a process that combines elements of both quantum computing and classical simulated annealing. It uses a quantum computer to solve optimization problems by gradually changing the set of qubits from an initial state to a final state while allowing for quantum fluctuations during the transition.

Advantageously, classical Ising Machine operations can be used as another technique in the method. These operations are based on the Ising model, a mathematical model of ferromagnetism that can be used to represent and solve combinatorial optimization problems using a set of qubits.

One of the technical advantages of the method lies in its ability to efficiently solve complex combinatorial optimization problems by leveraging the power of quantum computing and classical optimization techniques. By being configured to use one or several techniques, the method can find the global minimum solution for a given problem more effectively than traditional methods.

According to an embodiment, and as illustrated by figure 7, the present invention also relates to a computerized system 200. This system 200 is preferably configured to execute the method 100 previously described. Advantageously, the system 200 is configured to solve a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian.

According to an embodiment, the system comprises at least:

- A classical processing unit 201;
- A electromagnetic pulse generation module 202;
- A measurement module 203;
- A quantum processing unit 204;
- A communication module 205; and

- A storage module 206.

According to an embodiment, the classical processing unit 201 is configured to execute various tasks related to the problem encoding, state analysis, bias-field construction, Hamiltonian manipulation, and solution determination, for example. The classical processing unit 201 performs the following functions:

- Selection of at least one combinatorial optimization problem: The classical processing unit 201 selects a specific combinatorial optimization problem from a set of available problems.
- Encoding of the selected optimization problem into a final Hamiltonian: The classical processing unit 201 translates the chosen optimization problem into a form suitable for quantum computation, resulting in a final Hamiltonian that encodes the solution to the problem.
- Post-processing of state probability distributions and generation of energy probability distributions: The classical processing unit 201 processes the measured state probabilities of the qubits and generates corresponding energy probability distributions.
- Calculation of expectation values from energy probability distributions: The classical processing unit 201 calculates the expectation value of the Hamiltonian using the generated energy probability distributions.
- Extraction of ground state solutions and construction of bias-field Hamiltonians: The classical processing unit 201 identifies the ground state solution(s) from the calculated expectation values and constructs a bias-field Hamiltonian based on this information.
- Comparison of calculated energy levels and update of the initial Hamiltonian: The classical processing unit 201 compares the calculated energy levels with the desired energy level (e.g., the ground state energy) and updates the initial Hamiltonian accordingly to improve the solution quality in subsequent iterations.
- Determination of the solution to the combinatorial optimization problem: Based on the best outcome of the iterations, the classical processing unit 201 determines the final solution to the combinatorial optimization problem.

From a hardware perspective, the classical processing unit 201, also called central processing unit, is the primary component responsible for executing instructions in a

classical computer system. From a hardware perspective, the classical processing unit 201 is an electronic circuit designed to execute instructions and perform data processing tasks.

At its core, the classical processing unit 201 may comprise multiple interconnected units, each serving a specific function:

1. **Arithmetic Logic Unit (ALU):** The ALU performs arithmetic operations (addition, subtraction, multiplication, division) and logical operations (AND, OR, NOT, XOR) on binary data. It receives operands from registers and produces results that are stored back in the registers.
2. **Control Unit (CU):** The CU fetches instructions from memory, decodes them to understand what operation needs to be performed, and then orchestrates the flow of data between the various components of the classical processing unit 201. It also manages the program counter, which keeps track of the address of the next instruction to be executed.
3. **Registers:** Registers are high-speed memory areas within the classical processing unit 201 used for temporary storage of data and addresses. They serve as intermediaries between the ALU and other functional units, allowing efficient data transfer and manipulation. Examples include program counter (PC), memory address register (MAR), memory buffer register (MBR), accumulator (ACC), and general-purpose registers.
4. **Cache:** Modern classical processing units employ cache memory to store frequently accessed or recently used data and instructions, enabling faster access than main memory (RAM). Cache is organized into levels (L1, L2, L3, etc.), with each level having different sizes and speeds, but generally decreasing in size and increasing in speed from higher to lower levels.
5. **Bus Interface:** The classical processing unit 201 communicates with other components, units or modules of the system 200 through a shared bus. The bus interface unit manages data transfer between the classical processing unit 201 and these external components units, or modules.
6. **Clock:** A clock signal synchronizes the operations within the classical processing unit 201, ensuring that all components work in harmony to execute instructions in a sequential manner.

According to an embodiment, the electromagnetic pulse generation module 202 is configured to initialize and drive the set of qubits under the total Hamiltonian using time-dependent electromagnetic pulses. The electromagnetic pulse generation module 202 may perform the following functions:

- Emission of a plurality of electromagnetic pulses to initialize qubits into predetermined quantum states: The electromagnetic pulse generation module 202 is configured to emit a series of electromagnetic pulses to set the initial state of the qubits according to the chosen initialization scheme.
- Driving of the set of qubits under the total Hamiltonian using time-dependent electromagnetic pulses: The electromagnetic pulse generation module 202 is configured to apply a sequence of time-dependent electromagnetic pulses to evolve the quantum system according to the total Hamiltonian, which includes both the problem Hamiltonian and any auxiliary Hamiltonians (e.g., bias-field Hamiltonians).

From a hardware perspective, the electromagnetic pulse generation module 202 may comprise several hardware components working together to generate and apply predetermined electromagnetic pulses:

- Pulse Shaping Circuitry: This circuitry generates predetermined pulse shapes, amplitudes, frequencies, and phases required for initializing and driving the qubits according to the total Hamiltonian. It can comprise:
 - Pulse generators: Devices that create the basic pulse shapes (e.g., Gaussian, rectangular, or sinusoidal) with adjustable amplitude, duration, and phase.
 - Waveform memories: Storage devices that hold the pre-calculated pulse sequences for each qubit in the system.
 - Attenuators and amplifiers: Circuitry to adjust the power level of the pulses to match the specific requirements of the qubits and avoid excessive heating or damage.
- Frequency Synthesis and Control: To generate accurate electromagnetic pulses, the EPGM requires precise frequency references and control mechanisms:

- Frequency synthesizers: Devices that produce stable, tunable frequencies based on a master clock reference.
 - Phase-locked loops (PLLs): Circuitry that ensures the generated pulses maintain a fixed phase relationship with the qubits' internal oscillations.
- Pulse Distribution Network: This network routes the generated electromagnetic pulses to the appropriate qubits in the quantum processing unit 204. It can comprise:
 - Waveguides or transmission lines: Physical pathways for transporting microwave signals from the electromagnetic pulse generation module 202 to the quantum processing unit 204.
 - Switches and attenuators: Devices that control the routing, amplitude, and phase of the pulses to individual qubits or groups of qubits.
- Feedback Control System: To ensure accurate pulse application and minimize errors, the electromagnetic pulse generation module 202 can incorporate a feedback control system:
 - Error detection and correction circuits: Components that monitor the pulse generation process and apply corrections when deviations from the desired pulse parameters are detected.
 - Adaptive control algorithms: Software or hardware implementations of algorithms (e.g., proportional-integral-derivative controllers) that adjust pulse parameters in real-time based on feedback signals.

Preferably, the hardware components of the electromagnetic pulse generation module 202 are configured to work in concert with other modules, such as the classical processing unit 201, the quantum processing unit 204, and the measurement module 203, for example, to perform various actions, such as quantum annealing or adiabatic computing process.

According to an embodiment, the measurement module 203 is configured to measure the quantum state of each qubit in the set of qubits to obtain state probability distributions. The measurement module 203 can be configured to perform measurement of the quantum state of each qubit of a set of qubits to obtain state probability distributions. The measurement module 203 measures, for example in collaboration with the electromagnetic pulse generation module 202, the state of each

qubit and generates corresponding state probability distributions, which are then processed by the classical processing unit 201 for further analysis.

Preferably, the measurement module 203 may comprise several hardware components working together to measure the quantum state of each qubit and obtain state probability distributions:

- **Readout Circuits:** Readout circuits are responsible for detecting the final state of each qubit in the set of qubits. They can comprise, for example:
 - **Resonators:** Microwave cavities or transmission lines that couple to the qubits' transition frequencies, enabling sensitive detection of the qubits' states.
 - **Amplifiers:** Low-noise amplifiers that amplify the weak signals from the resonators without introducing excessive noise.
 - **Attenuators:** Circuitry to control the signal level and prevent saturation or damage to downstream components.
- **Detection Electronics:** Detection electronics process the amplified signals to extract information about the qubits' states:
 - **Mixers:** Devices that downconvert the amplified signals from the readout frequency to an intermediate frequency or baseband for further processing.
 - **Analog-to-Digital Converters:** Circuitry that converts the analog signals into digital data streams for analysis by the classical processing unit 201.
- **Data Acquisition System:** The data acquisition system manages the collection and transfer of measurement data from the detection electronics to the classical processing unit 201:
 - **Data buffers:** Temporary storage areas for incoming measurement data.
 - **Data interfaces:** Hardware interfaces (e.g., USB, PCIe, or custom high-speed serial links) that facilitate data transfer between the measurement module 203 and the classical processing unit 201.
- **Feedback Control System:** To ensure accurate and reliable measurements, the measurement module 203 can comprise a feedback control system:
 - **Error detection and correction circuits:** Components that monitor the measurement process and apply corrections when deviations from the desired performance are detected.

- Adaptive control algorithms: Software or hardware implementations of algorithms (e.g., proportional-integral-derivative controllers) that adjust measurement parameters in real-time based on feedback signals.

As previously described, one of the primary functions of the measurement module 203 is to measure the quantum state of each qubit in the set of qubits, generating corresponding state probability distributions. This process typically involves:

- Readout: The readout circuits detect the final state of each qubit by applying an electromagnetic pulse, using the electromagnetic pulse generation module 202, at the qubit's transition frequency and measuring the reflected or transmitted signal.
- Data Acquisition: The detection electronics process the signals from the readout circuits, converting them into digital data streams that can be analyzed by the classical processing unit 201.
- State Probability Distribution: The measurement module 203 generates state probability distributions by counting the occurrences of each unique bitstring (i.e., combination of qubit states) in a series of measurements. Each bitstring is associated with a probability value, representing the number of times it was measured relative to the total number of measurements.

Advantageously, the hardware components of the measurement module 203 are configured to work in concert with other modules, such as the electromagnetic pulse generation module 202, the quantum processing unit 204 and the classical processing unit 201, for example, to perform various actions.

According to an embodiment, the quantum processing unit 204 is configured to perform quantum computations. The quantum processing unit 204 comprises a set of qubits and a set of qubit gates, comprising single and/or multi-qubit gates. The quantum processing unit 204 is configured to perform the following functions:

- Initialization of the set of qubits in an initial configuration corresponding to the ground state of the initial Hamiltonian: The quantum processing unit 204 initializes the quantum system in the ground state of the initial Hamiltonian using appropriate initialization techniques such as adiabatic state preparation or counterdiabatic state preparation.

- Evolution of the initial Hamiltonian according to the total Hamiltonian to reach at least one low energy state of the final Hamiltonian: The quantum processing unit 204 applies a sequence of quantum gates and electromagnetic pulses to evolve the quantum system under the total Hamiltonian, aiming to reach a low-energy state of the final Hamiltonian that encodes the solution to the problem.
- Provision of measured states of the qubits in a computational basis: The quantum processing unit 204 provides the measured states of the qubits in a computational basis for further analysis by the classical processing unit 201.

From a hardware perspective, the quantum processing unit 204 comprises several hardware components working together to create, manipulate, and measure qubits states:

- Qubits: One of the fundamental elements of quantum information in the quantum processing unit 204 are qubits, which can exist in multiple states simultaneously due to the principles of superposition. Qubits can be realized using, for example:
 - Superconducting circuits: Qubits based on superconducting circuits use Josephson junctions and capacitive or inductive elements to create artificial atoms with well-defined energy levels.
 - Trapped ions: In trapped ion qubits, individual atomic ions are confined and manipulated using electromagnetic fields. The internal electronic states of the ions serve as qubit states.
 - Semiconductor quantum dots: Qubits based on semiconductor quantum dots use the spin or charge degrees of freedom of electrons confined in potential wells created by electrostatic gates.
- Qubit Gates: Qubit gates are the building blocks for implementing quantum algorithms and manipulating qubits' states. They can be categorized into:
 - Single-qubit gates: Operations that act on a single qubit, such as Pauli gates (X, Y, Z), Hadamard gate (H), phase shift gates (S, T), and their combinations.
 - Multi-qubit gates: Operations that involve multiple qubits, such as controlled-NOT (CNOT) gate, controlled-Z (CZ) gate, and other entangling gates like the Toffoli or Fredkin gate.

- **Control Electronics:** The control electronics manage the application of quantum gates and pulses to manipulate qubit states. For example, the quantum processing unit 204 can comprise the electromagnetic pulse generation module 202.
- **Measurement Electronics:** The measurement electronics enable the detection of qubit states. For example, the quantum processing unit 204 can comprise the measurement module 203.
- **Qubit Array and Connectivity:** The physical arrangement of qubits in the quantum processing unit 204 is useful for implementing specific algorithms and solving particular problems:
 - **Qubit array:** An array of qubits arranged in a grid or other topology, with each qubit connected to its neighbors through qubit gates, for example.
 - **Connectivity:** The pattern of connections between qubits, which determines the types of interactions and entanglement that can be created between them.

Preferably, the hardware components of the quantum processing unit 204 work in concert with other modules, such as the electromagnetic pulse generation module 202, the measurement module 203, and the classical processing unit 201, for example, to perform various actions.

According to an embodiment, the communication module 205 is configured to exchange information between the various components of the system and between at least one user and the system through for example a user interface.

Preferably, the communication module 205 can serve as the intermediary between the classical processing unit 201, the quantum processing unit 204, and other external devices, units, modules or systems. It facilitates the exchange of data, control signals, and synchronization information between these components to ensure seamless coordination.

From a hardware perspective, the communication module 205 can comprise several hardware components responsible for managing data flow and synchronization:

- **Data Interfaces:** The communication module 205 incorporates various data interfaces to connect with different hardware components:

- Quantum processing unit interface: A high-speed, low-latency interface (e.g., custom serial link, PCIe, or optical interconnect) that enables the exchange of quantum state information, control signals, and synchronization pulses between the classical processing unit 201 and the quantum processing unit 204.
 - Classical data interfaces: Interfaces (e.g., USB, Ethernet, Wi-Fi, or Bluetooth) for connecting external devices such as user interface devices, storage devices, or other classical processing units.
- Buffer Memories: The communication module 205 can comprise buffer memories to temporarily store incoming or outgoing data:
 - Input buffers: Temporary storage areas for data received from external sources via the classical data interfaces.
 - Output buffers: Temporary storage areas for data destined for external recipients, such as the quantum processing unit 204 or other connected devices, units or modules.
- Synchronization Circuits: The communication module 205 incorporates synchronization circuits to ensure proper timing and coordination between hardware components:
 - Clock distribution: Circuitry that generates and distributes clock signals to synchronize the operation of various hardware components within the communication module 205 and between the communication module 205, the classical processing unit 201, and the quantum processing unit 204.
 - Trigger generation: Circuits that generate trigger signals to initiate specific actions or processes at precise moments in time, such as starting a quantum algorithm or capturing measurement results.
- Control Logic: The communication module 205 can house control logic responsible for managing data flow, synchronization, and error handling:
 - Data routing: Control circuits that direct incoming data to the appropriate destinations within the communication module 205 or other connected hardware components.
 - Error detection and correction: Circuits that monitor data integrity and implement error detection (e.g., parity checks, cyclic redundancy

checks) and correction techniques (e.g., forward error correction) to ensure reliable communication.

Preferably, the hardware components of the communication module work in concert with other modules, such as the classical processing unit 201 and the quantum processing unit 204, to perform various actions.

According to an embodiment, the storage module 206 is configured to store each iteration's ground state solution associated with its calculated energy level for future reference or analysis.

Preferably, the storage module 206 is configured to serve as the memory component responsible for storing classical data, processor-executable instructions, and other information required for the operation of the method 100. It facilitates efficient access to and retrieval of data by the classical processing unit 201.

From a hardware perspective, the storage module 206 can comprise various hardware components that enable reliable data storage and retrieval:

- **Volatile Memory:** The storage module 206 can comprise volatile memory devices for temporary storage of dynamic data:
 - **Random Access Memory (RAM):** A type of volatile memory used for short-term data storage, allowing fast access to data by the classical processing unit 201 during algorithm execution.
- **Non-Volatile Memory:** The storage module 206 incorporates non-volatile memory devices for long-term data storage and persistence:
 - **Read-Only Memory (ROM):** Non-volatile memory used for storing firmware or bootloader software that initializes the system and loads the operating system into RAM upon startup.
 - **Flash Memory:** Reprogrammable, non-volatile memory that retains stored information even when power is turned off. Flash memory can be used to store operating systems, applications, or user data.
 - **Solid State Drives (SSDs) / Hard Disk Drives (HDDs):** Non-volatile storage devices that provide large-capacity data storage for the system. SSDs use flash memory technology, while HDDs rely on rotating platters and magnetic heads to read and write data.

- **Storage Controllers:** The storage module 206 features storage controllers responsible for managing data flow between the classical processing unit 201 and various storage devices:
 - **Memory Controller:** A hardware component that manages data transfer between the classical processing unit 201 and memory devices (RAM, ROM, etc.).
 - **Disk Controller:** A hardware component that controls data access to non-volatile storage devices such as SSDs or HDDs.
- **Data Buffers:** The storage module 206 includes data buffers to temporarily store incoming or outgoing data:
 - **Input buffers:** Temporary storage areas for data received from external sources via the communication module 205 or other connected components.
 - **Output buffers:** Temporary storage areas for data destined for external recipients, such as the classical processing unit 201, the communication module 205, or other connected devices, units or modules.

Preferably, the hardware components of the storage module 206 are configured to work in concert with other modules, such as the classical processing unit 201 and the communication module 205, for example, to perform various actions.

The system can be further enhanced by incorporating additional features such as error correction mechanisms, noise mitigation techniques, and adaptive learning algorithms to improve the overall performance and robustness of the system. The system can also be integrated with classical optimization algorithms, classical Ising machines, or other quantum computing devices to leverage their unique capabilities and enhance the solution quality.

In summary, the present invention provides a novel method for optimizing combinatorial problems using both analog and digital quantum computing techniques. The method integrates solutions from different hardware platforms through bias terms, enabling the enhancement of solutions through successive iterations using the bias (anti-bias)-field derived from previous solutions to guide the computation towards better solutions. The present invention can be employed to enhance the solutions

derived from existing classical heuristic algorithms and improve the solutions obtained from classical Ising machines. The present invention significantly outperforms traditional DCQO and finite-time adiabatic quantum optimization, as well as the Quantum Approximate Optimization Algorithm (QAOA), in success probability and approximation ratio. Experimental implementation on various quantum processors has demonstrated its effectiveness for solving large-scale combinatorial optimization problems.

The present invention introduces a novel method for solving combinatorial optimization problems using both analog and digital quantum computing techniques. This method, called bias field digitized counterdiabatic quantum optimization (bf-DCQO), iteratively refines the Hamiltonian with information from prior solutions to improve the likelihood of ground state convergence. The invention provides significant advantages over existing optimization techniques, both quantum and classical, including enhanced convergence rates, reduced susceptibility to local minima, and improved computational efficiency.

Examples of implementation of the present invention:

1) Ising spin-glass problem with long-range interactions:

To analyze the performance of the bf-DCQO, consider 400 random instances of the spin-glass problem have been considered, with coupling J_{ij} and h_i obtained from a Gaussian distribution with a mean of 0 and a variance of 1. The scheduling function is $\lambda(t) = \sin^2[\frac{\pi}{2} \sin^2(\frac{\pi t}{2T})]$, and $h_i^x = -1$ is considered. The inventors only consider first-order CD terms, and the CD coefficient changes during each iteration since the initial Hamiltonian changes. Here, we analytically calculate the exact form of the CD coefficient α_1 for the Hamiltonian in Eq. 2. Fig. 3 (b) illustrates the ground state success probability with increasing system size for bf-DCQO with 10 iterations and naive DCQO, given a fixed evolution time T with three Trotter steps. In both cases, the success probability $p_{gs} = |\langle \psi_{gs} | \psi_f(T) \rangle|^2$, where $|\psi_f(T)\rangle$ is the actual ground state of the spin-glass Hamiltonian, decreases exponentially with system size. However, the exponential factor for bf-DCQO is smaller than that for DCQO, indicating a polynomial scaling advantage. We noticed that not all 400 instances are improved by the inclusion

of the bias field. In some cases, if the solution from the first iteration of DCQO leads to undesired outcomes, employing an anti-bias field $h_i^b = -\langle \sigma_i^z \rangle$, may help suppress these outcomes. In Fig. 3 (c), the number of instances enhanced by the bias field is depicted. For the unsuccessful instances, employing an anti-bias field shows improvement. However, there are few instances where both the bias and anti-bias fields fail. This is primarily due to the simulation parameters we have chosen in this work, and altering them may lead to successful results.

2) Noisy simulation:

To evaluate the performance of the algorithm in the presence of hardware noise, we use a noisy emulator that mimics the actual noise model of a trapped-ion hardware system, IonQ Forte. We consider a fully connected 29-qubit spin-glass instance and implement the bf-DCQO algorithm. The energy distribution across each iteration is shown in Fig. 3 (d). Remarkably, even with just 2 Trotter steps and the number of shots $n_{shots} = 1000$, the algorithm guides the dynamics toward the solution. By iteration 29, the exact ground state was obtained. Additionally, it is clear that the approximation ratio $\frac{E_{obtained}}{E_{gs}}$ improves with each iteration.

An important aspect of bf-DCQO is that it does not require any classical optimization subroutines as in variational quantum algorithms (VQA). This feature makes it an impressive approach, as the main drawback of VQA lies in trainability issues like barren plateaus and local minima. The presence of noise makes it even harder to rely on VQAs. Since we have already seen the successful performance of bf-DCQO in noisy conditions, we now compare the performance of bf-DCQO with a widely used variational quantum optimization algorithm, QAOA. We consider 10 random instances of the long-range spin-glass problem across various sizes. We use ground state success probability and approximation ratio as metrics for comparison.

To maintain the same circuit depth, we consider QAOA with 3 layers ($p = 3$) and bf-DCQO with 3 Trotter steps. For optimizing the QAOA circuit, we use the COBYLA optimizer with a maximum of 300 iterations. For each instance, the best solution out of 20 random initializations is considered for QAOA. For bf-DCQO, we employ just 10 iterations of feedback. In Fig. 4, we plot the success probability enhancement ratio, which is the ratio of the ground state success probability obtained with bf-DCQO

versus QAOA, as well as the approximation ratio enhancement ratio. Despite requiring two orders of magnitude fewer iterations, bf-DCQO outperforms QAOA in both metrics. Moreover, the success probability enhancement ratio increases with system size, showing a 75x improvement for the 20-qubit case. On average, we observe a 1.3x improvement in the approximation ratio with bf-DCQO.

3) Experimental implementation:

For the experimental validation of bf-DCQO, we consider a 36-qubit trapped-ion quantum processor, IonQ Forte, and a 127-qubit superconducting quantum processor, IBM Brisbane. We explore two problems that can be suitably mapped to the hardware connectivity: a randomly generated Weighted Maximum Independent Set (WMIS) problem with 36 nodes, implemented on trapped-ion hardware, and an instance of the Ising spin-glass problem on a heavy-hex lattice with 100 spins, implemented on superconducting hardware.

The WMIS is a combinatorial optimization problem where the objective is to identify a subset of vertices in a graph that are mutually non-adjacent (an independent set) and have the highest possible total weight. Given a graph $G = (V, E)$ with vertices V and edges E , each vertex $v \in V$ has an associated weight $w(v)$. The task is to find a subset of vertices $I \subseteq V$ such that no two vertices in I are connected by an edge in E , while maximizing the sum of the weights of the selected vertices:

$$\text{Maximize } \sum_{v \in I} w(v), \text{ subject to } (u, v) \notin E \ \forall u, v \in I.$$

This problem is NP-hard because it generalizes the classic MIS problem by incorporating vertex weights. The WMIS problem can be mapped to the Ising spin-glass Hamiltonian by associating a binary spin variable with each vertex, and defining interactions that penalize adjacent vertices that are both included, while rewarding vertices that are selected based on their weights.

Since in the WMIS problem the interactions between the nodes/qubits can be long-range, trapped-ion systems are well suited to tackle this problem without requiring any SWAP gates. In Fig. 5 (a), the experimental result from IonQ Forte for a 36-node WMIS is shown. Due to limited access to the hardware, we first ran the bf-DCQO on an ideal local simulator and then ran only the final circuit corresponding to the last iteration on the hardware. The error-mitigated experimental result is in close agreement with the

ideal simulation result. In the experiment, we considered 3 Trotter steps, 2500 shots, and used native gates for circuit implementation. Additionally, we performed debias error mitigation and circuit optimization to reduce the total gate counts further. For the considered WMIS problem, the maximum independent set size is 16, and the obtained independent set size is 11.

As a second example, a spin-glass problem on a heavy hexagonal lattice is considered. Since the interaction terms in the problem Hamiltonian match the hardware connectivity, we can consider a large system size of 100 qubits on IBM Brisbane hardware. In Fig. 5 (b), we show the ideal simulation results for DCQO and bf-DCQO, and the experimental result for bf-DCQO. We also consider a classical solver, Gurobi, as a reference. We notice that, even with just 10 iterations, bf-DCQO provides a drastic enhancement compared to DCQO. Additionally, in the absence of noise, bf-DCQO reaches the solution obtained from Gurobi with just two Trotter steps. Although the experimental results are slightly different from the ideal result due to noise, the performance is better than the ideal DCQO.

In the present description, the words “data”, “piece of information” and “information” can be used for the same purpose, i.e. one data can be or comprise information or a piece of information, and a piece of information can be or comprise a data.

In the present description, one module can comprise several modules, one module can be formed of several modules. For example, the quantum channel module can comprise several other modules and quantum channels.

According to an embodiment, the present invention can be configured to cooperate with at least one input module including one or more user interface devices such as a keyboard, pointer, number pad, or touch screen.

According to an embodiment the classical information processing module comprises at least one processor. In the present description, a processor may be any logic processing unit, such as one or more digital processors, microprocessors, central processing units, graphics processing units, application-specific integrated circuits, programmable gate arrays, programmed logic units, digital signal processors, network processors, and the like.

In the present description, a storage device is at least one non-transitory or tangible storage device. A storage device can, for example, include one or more volatile storage devices, for instance random access memory, and one or more non-volatile storage devices, for instance read only memory, flash memory, magnetic hard disk, optical disk, solid state disk, and the like.

In the present description, a storage device can include or store processor-executable instructions and/or processor-readable data associated with the operation of the present invention and/or with the execution of the method of the present invention. Execution of processor-executable instructions and/or data causes the at least one processor, and/or control modules or units, to carry out various processes and actions.

Classical data can include processor-executable instructions that, when executed by a processor, cause the processor to control, initialize, write to, manipulate, read out, and/or otherwise send data to/from a quantum module.

Classical data can include data used or obtained by the operation of the present invention. Classical data can include data associated with, e.g., created by, referred to, changed by, a processor executing processor-executable instructions, such as, control instructions.

Quantum data or quantum information can comprise one or more qubits. A qubit or quantum bit is a logical building block of a quantum computer comparable to a binary digit in a classical digital computer. A qubit conventionally is a defined physical system having two or more discrete states called computational states or basis states. Basis states logically are analogous to binary states. These states may be labeled $|0\rangle$ and $|1\rangle$.

As well known by the skilled person in the relevant art (see for example “Quantum entanglement”, Ryszard Horodecki et al., Rev. Mod. Phys. 81, 865), an entangled state corresponds to at least two systems, for example two particles, sharing one wave function, and where the determination of a property of one of the particles determines a property of the other particle.

Unless otherwise specified herein, or unless the context clearly dictates otherwise the term about modifying a numerical quantity means plus or minus ten percent. Unless

otherwise specified, or unless the context dictates otherwise, between two numerical values is to be read as between and including the two numerical values.

In the present description, some specific details are included to provide an understanding of various disclosed implementations. The skilled person in the relevant art, however, will recognize that implementations may be practiced without one or more of these specific details, parts of a method, components, materials, etc. In some instances, well-known methods associated with quantum computing, quantum algorithms, quantum processes, artificial intelligence, machine learning and/or neural networks, have not been shown or described in detail to avoid unnecessarily obscuring descriptions of the disclosed implementations. In some instances, well-known structures associated with semiconductor and/or optical devices and/or quantum computing and/or quantum information processing, such as targets, substrates, lenses, waveguides, shields, filters, lasers, processor-executable instructions, have not been shown or described in detail to avoid unnecessarily obscuring descriptions of the disclosed implementations.

In the present description and appended claims "a", "an", "one", or "another" applied to "embodiment", "example", or "implementation" is used in the sense that a particular referent feature, structure, or characteristic described in connection with the embodiment, example, or implementation is included in at least one embodiment, example, or implementation. Thus, phrases like "in one embodiment", "in an embodiment", or "another embodiment" are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be combined in any suitable manner in one or more embodiments, examples, or implementations.

As used in this description and the appended claims, the singular forms of articles, such as "a", "an", and "the", may include plural referents unless the context mandates otherwise. Unless the context requires otherwise, throughout this description and appended claims, the word "comprise" and variations thereof, such as, "comprises" and "comprising" are to be interpreted in an open, inclusive sense, that is, as "including, but not limited to".

Modifications and improvements to the above-described implementations of the present invention may become apparent to those skilled in the art. The foregoing

description is intended to be exemplary rather than limiting. The scope of the present invention is, therefore, intended to be limited solely by the scope of the appended claims.

Claims

1. A computer-implemented method (100) for solving a combinatorial optimization problem, the solution of which is encoded in at least one energy state of a Hamiltonian, the method (100) being configured to be implemented by at least one computerized system (200) comprising at least one classical processing unit (201), at least one electromagnetic pulse generation module (202), at least one measurement module (203) and at least one quantum processing unit (204), the quantum processing unit (204) comprising a plurality of qubits and a plurality of qubit gates, the method (100) comprising at least the following steps:
 - a) Selecting (101), using at least one classical processing unit (201), at least one combinatorial optimization problem;
 - b) Providing (102), using at least one communication module (205), an initial Hamiltonian comprising an initial ground state, said initial ground state being configured to be computationally initialized over at least one set of qubits from a plurality of qubits of a quantum processing unit (204), the initialization of the ground state being done using at least one state-preparation process, the preparation process being configured to use at least one set of qubit gates from a plurality of qubit gates of the quantum processing unit (204) to manipulate the quantum state of the set of qubits using a set of electromagnetic pulses, using an electromagnetic pulse generation module (202);
 - c) Providing (103), using the communication module (205), a final Hamiltonian configured to encode the selected optimization problem, the final Hamiltonian comprising a set of energy states comprising a final energy state, the final energy state being configured to encode a solution of the combinatorial optimization problem;
 - d) Encoding (104), using the classical processing unit (201), the selected optimization problem to the final Hamiltonian;
 - e) Iterating (105) the following steps until the final energy state is reached based on a bias-field feedback mechanism:
 - Initializing, using the quantum processing unit (204) and the electromagnetic pulse generation module (203), the set of qubits in an initial configuration, the initial configuration comprising initial

quantum states for each qubit of the set of qubits and initial interactions between each qubit of the set of qubits, the initial configuration of the set of qubits corresponding to the ground state of the initial Hamiltonian, this initialization step comprising at least:

- i. Putting each qubit of the set of qubits into predetermined quantum states by emitting a plurality of electromagnetic pulses using the electromagnetic pulse generation module (202);
 - ii. Constructing a time-dependent adiabatic Hamiltonian comprising a summation of the initial Hamiltonian and the final Hamiltonian with time-dependent control parameters, the time-dependent control parameters comprising a set of control electromagnetic pulses configured to drive the initial Hamiltonian to the final Hamiltonian using the electromagnetic pulse generation module (202);
 - iii. Applying a series of operator transformations to the time-dependent adiabatic Hamiltonian, the operator transformations comprising at least one recursive series of nested commutators, each commutator of the series of nested commutators involving at least one adiabatic Hamiltonian and its derivative with respect to the time-dependent control parameters;
 - iv. Modifying the terms of the adiabatic Hamiltonian by incorporating contributions from the nested commutators to generate at least one counter-diabatic Hamiltonian;
 - v. Summing the adiabatic Hamiltonian with the counter-diabatic Hamiltonian to obtain a total Hamiltonian;
- Evolving the quantum state of the set of qubits according to the total Hamiltonian to reach at least one of the lowest energy states of the final Hamiltonian, the evolving step comprising:
 - i. Counter-diabatically evolving the set of qubits with the total Hamiltonian up to a predetermined final time which is equal to or less than the coherence time of the set of qubits, this evolving step comprising:

1. Emitting a set of time-dependent electromagnetic pulses, using the electromagnetic pulse generation module (202), to drive the set of qubits under the total Hamiltonian, the set of time-dependent electromagnetic pulses being used to generate interactions among the qubits of the set of qubits according to the total Hamiltonian;
- Measuring, using at least one measurement module (203), each qubit of the set of qubits to obtain at least one state probability distribution, the state probability distribution containing a set of pairs of unique bitstring and its associated probability, the unique bitstring being the measured states of the qubits in a computational basis, the probability being the number of times the unique bitstring was measured relative to the total number of total measurements;
 - Post processing, using the classical processing unit (201), the result of the state probability distribution, the post processing comprising the following steps:
 - i. Building an energy probability distribution as a set of triplets, each triplet containing one unique bitstring, one energy value, which is the expectation value of the unique bitstring with respect to the final Hamiltonian, and the probability associated to the bitstring and retrieved from the state probability distribution;
 - ii. Generating a post-processed energy probability distribution by performing a mapping of the probabilities of each triplet over a predetermined scale;
 - Calculating, using the classical processing unit (201), at least one expectation value from the post-processed energy probability distribution;
 - Extracting, using the classical processing unit (201), a ground state solution from the calculated expectation value;
 - Constructing, using the classical processing unit (201), a bias-field Hamiltonian from the final ground state by performing a product of

the calculated expectation values corresponding to each qubit and operator transformation, and by performing a summation of a plurality of local bias-field Hamiltonians, each local bias-field Hamiltonian of the plurality of local bias-field Hamiltonians being obtained from at least one qubit of the set of qubits;

- Calculating, using the classical processing unit (201), the energy level of the ground state solution;
 - Saving, in at least one storage module (206), the calculated energy level to a list comprising each iteration of ground state solution associated with its calculated energy level;
 - Comparing, using the classical processing unit (201), the calculated energy level with the other calculated energy levels from the list, if the calculated energy level is equal to the lowest energy level already saved within the list, stopping the iterating step (105);
 - Updating, using the classical processing unit (201), the initial Hamiltonian by summing the initial Hamiltonian with the constructed bias-field Hamiltonian;
- f) Obtaining (106), using the classical processing unit (201), the calculated expectation value from the last ground state solution;
- g) Determining (107), using the classical processing unit (201), the solution to the combinatorial optimization problem as being the identified expectation value.

2. The method (100) according to the previous claim wherein the post-processing further comprises, before the step of generating a post-processed energy probability distribution, at least the following steps:
- Sorting the energy probability distribution;
 - Removing at least one portion of the energy probability distribution, said removed portion corresponding to triplets having an energy value higher than the remaining portion of the energy probability distribution to reduce the energy probability distribution, the remaining energy probability distribution comprising at least one triplet having the lowest energy value over the whole energy probability distribution;

- On the reduced energy probability distribution, performing at least one local search on each unique bitstring, where the local search comprises qubit gate operations on the qubits of the set of qubits to find a new unique bitstring having a lower energy than the unique bitstring with the lowest energy within the reduced energy probability distribution, if the new unique bitstring is found, the triplet corresponding to the unique bitstring with the lowest energy is replaced in the energy probability distribution.
3. The method (100) according to claim 2, wherein the remaining energy bitstrings have larger probabilities.
 4. The method (100) according to any one of claims 1 to 3, wherein the electromagnetic pulses are global or local electromagnetic pulses regarding the set of qubits.
 5. The method (100) according to any one of claims 1 to 4, wherein the constructed bias-field Hamiltonian is an anti-bias-field Hamiltonian.
 6. The method (100) according to any one of claim 1 to 5, wherein the state-preparation process is selected from the group consisting of: quantum annealing, adiabatic state preparation, and counter-diabatic state preparation.
 7. The method (100) according to any one of claim 1 to 6, comprising, before the iterating step (105), a step of summing the initial Hamiltonian with at least one generic bias-field Hamiltonian obtained by extracting a non-optimal solution of the optimization problem from a quantum or a classical algorithm running on a quantum or classical hardware.
 8. The method (100) according to any one of claim 1 to 7, wherein the constructed bias-field Hamiltonian is derived from solutions obtained using classical heuristic algorithms, quantum annealers, or iterative digitized-counterdiabatic quantum optimization.

9. The method (100) according to any one of claim 1 to 8, wherein the counter-diabatic Hamiltonian is based on a biased or anti-biased digitized-counter-diabatic protocol.
10. The method (100) according to any one of claim 1 to 7, wherein the ground state solution is obtained using one or more techniques selected from the group consisting of: adiabatic quantum optimization, quantum annealing, simulated annealing, simulated quantum annealing, and classical Ising Machine operations.
11. A computerized system (200) configured to solve a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian, the system (200) comprising:
- at least one classical processing unit (201) configured to:
 - select at least one combinatorial optimization problem;
 - encode the selected optimization problem into a final Hamiltonian;
 - post-process state probability distributions and generate energy probability distributions;
 - calculate expectation values from energy probability distributions;
 - extract ground state solutions and construct bias-field Hamiltonians;
 - compare calculated energy levels and update the initial Hamiltonian; and
 - determine the solution to the combinatorial optimization problem;
 - at least one electromagnetic pulse generation module (202) configured to:
 - emit a plurality of electromagnetic pulses to initialize qubits into predetermined quantum states;
 - drive the set of qubits under the total Hamiltonian using time-dependent electromagnetic pulses;

- at least one measurement module (203) configured to measure the quantum state of each qubit of a set of qubits to obtain state probability distributions;
- a quantum processing unit (204) comprising:
 - a plurality of qubits; and
 - a plurality of qubit gates, preferably including single or multi-qubit gates,

wherein the quantum processing unit (204) is configured to:

- initialize the set of qubits in an initial configuration corresponding to the ground state of the initial Hamiltonian;
 - evolve the initial Hamiltonian according to the total Hamiltonian to reach at least one low energy state of the final Hamiltonian; and
 - provide measured states of the qubits in a computational basis;
- at least one communication module (205) configured to provide the initial Hamiltonian, the final Hamiltonian;
 - at least one storage module (206) configured to store each iteration's ground state solution associated with its calculated energy level.

12. A computer product program for solving a combinatorial optimization problem encoded in at least one energy state of a Hamiltonian which, when executed by at least one computerized system (200) according to claim 11, executes the method (100) according to any one of claims 1 to 10.

13. A non-transitory computer readable medium comprising at least one computer program product according to claim 12.