(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 December 2021 (02.12.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/237350 A1

(51) International Patent Classification:
G06F 17/10 (2006.01) *G06F 3/048* (2013.01)
G06F 17/11 (2006.01)

(21) International Application Number:
PCT/CA2021/050709

(22) International Filing Date:
26 May 2021 (26.05.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/030,803 27 May 2020 (27.05.2020) US

(71) Applicant: **1QB INFORMATION TECHNOLOGIES INC.** [CA/CA]; #200 - 1285 West Pender Street, Vancouver, British Columbia V6E 4B1 (CA).

(72) Inventors: **KUTTIMALAI, Silvan Shiwa**; #200-1285 West Pender St., Vancouver, British Columbia V6E 4B1 (CA). **ARAMON BAJESTANI, Maliheh**; #200-1285 West Pender St., Vancouver, BC V6E 4B1, Canada, Vancouver, British Columbia V6E 4B1 (CA). **ROSENBERG, Gilad Amir**; #200-1285 West Pender St., Vancouver, British Columbia V6E 4B1 (CA). **ROSE, Nathan Andrew**; #200-1285 West Pender St., Vancouver, British Columbia V6E 4B1 (CA). **WOODS, Bradley David**; #200-1285 West Pender St., Vancouver, British Columbia V6E 4B1 (CA).

(74) Agent: **SCHROEDER, Hans** et al.; Gowling WLG (CANADA) LLP, Suite 2600, 160 Elgin Street, Ottawa, Ontario K1P 1C3 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:
— *with international search report (Art. 21(3))*
— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

(54) **Title:** METHODS AND SYSTEMS FOR SOLVING AN OPTIMIZATION PROBLEM USING A FLEXIBLE MODULAR APPROACH

(57) **Abstract:** A method for solving an optimization problem inputted by a user may include: (a) using an interface of a digital computer to receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms; (b) using said one or more computational platforms to execute said one or more algorithms to yield a solution to said optimization problem; and (c) providing said solution to said optimization problem from said computational platform at said interface.

WO 2021/237350 A1

METHODS AND SYSTEMS FOR SOLVING AN OPTIMIZATION PROBLEM USING A FLEXIBLE MODULAR APPROACH

CROSS-REFERENCE

[0001] This application claims the benefit of U.S. Provisional Application No. 63/030,803, filed May 27, 2020, which application is incorporated herein by reference in its entirety.

BACKGROUND

[0002] The classical solvers may be designed to run on CPUs and their performance may have benefitted from the steady exponential increase in available computational power, in the last couple of decades. However, doubling the performance of CPUs every two years is expected to reach its physical limit in the current decade. Since real-world optimization problems may be NP-hard, both exact and heuristic methods may utilize exponential computational effort to solve an optimization problem as its size grows. Accordingly, many optimization problems may be intractable as the size of the optimization problem grows.

SUMMARY

[0003] General classical optimization techniques, for example, those embedded in modern solvers such as CPLEX, Gurobi, SCIP, and Baron, may be based on a search tree and may be used to solve optimization problems in various domains such as service, transportation, healthcare, energy, and finance. However, most search-tree based classical solvers are exact, that is, they may be designed to approach global optimality. In other cases, a classical solver may be a heuristic solver. A heuristic solver may be designed to find increasingly better solutions as the computational effort is increased, whereas exact solvers are designed to prove optimality, which may happen at the end of the process. In some cases, the early termination of an exact solver may turn it into an inefficient heuristic solver. For example, an exact solver may terminate before reaching an exact solution. The commercial exact solvers are general solvers and may not be designed to benefit from the specific structure of a given problem type. Therefore, even medium-sized instances may be intractable for exact solvers. However, heuristic methods may also be efficient for solving practically relevant optimization problems. This may be true even if or especially if they exploit the structure of the problem class being solved, such as, for example, by taking advantage of a formulation that more naturally represents the problem. Further, while problem specific knowledge may aid in convergence, in some instances, it may be impractical to rely on problem-specific knowledge to design custom solvers for every problem type.

[0004] Recognized herein is the need for improved methods and systems that will overcome at least one of the above-identified drawbacks. The present disclosure provides methods and systems for solving an optimization problem using a flexible modular approach. The present

disclosure may improve upon existing optimization solvers in at least some aspects. For example, the methods and systems disclosed herein may reduce the reliance on problem-specific knowledge to design custom solvers for every problem type. For example, the methods and systems disclosed herein may help to sustain the computational performance growth rate in the post-Moore's law era. Methods and systems disclosed herein may be designed to work in concert with new technologies such as quantum and application-specific computing. These new computing technologies may be fast, general-purpose, and power efficient; however, they have yet to demonstrate their efficiency in solving industry problems.

[0005] Methods and systems disclosed herein may use new computational hardware to solve problems such as quadratic unconstrained binary optimization (QUBO) problems. QUBO problems may be of particular interest because a large number of optimization problems can be recast into a QUBO form in a polynomial time. Methods and systems disclosed herein may map the solution space of the original problem into an unconstrained and binary space, thereby addressing an obstacle to the practical usage of special-purpose hardware. Methods and systems disclosed herein may address existing difficulties with mapping to an unconstrained and binary space such as: increasing irregularity of the objective function thereby increasing susceptibility to outputting local rather than global extrema. For example, turning a constrained optimization problem into a QUBO may be achieved by expanding the search space to include infeasible solutions, which are penalized in the objective function. However, this may expand the solution space significantly such that the ratio of infeasible solutions to feasible solutions is exponential. Furthermore, it makes the objective function highly irregular and leads to a search space with many local extrema. As a result, it may be significantly harder for the solver to find optimal or near-optimal solutions.

[0006] In an aspect, a method for solving an optimization problem inputted by a user is provided. The method may comprise: (a) using an interface of a digital computer to receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms; (b) transmitting a request to said one or more computational platforms to execute said one or more algorithms, wherein upon execution of said one or more algorithms by said one or more computational platforms said one or more computation platforms

yields a solution to said optimization problem; and (c) providing said solution to said optimization problem from said computational platform at said interface.

[0007] In some embodiments, said interface is a user interface (UI). In some embodiments, said UI is a graphical UI (GUI). In some embodiments, said one or more optimization problem specifications comprises at least one of: a search space comprising configurations of a plurality of search spaces, one or more objective functions of a plurality of objective functions, an allowed move of a plurality of allowed moves, or at least zero constraints of a plurality of constraints. In some embodiments, (a)(ii) comprises on said interface (1) presenting at least a subset of said plurality of optimization problem specifications to said user and (2) receiving a selection of said one or more optimization problem specifications from said user. In some embodiments, the method further comprises receiving a selection of said one or more optimization problem specifications from said user, wherein said selection comprises one or more of a selected search space comprising configurations, a selected allowed move, one or more selected objective functions, and at least zero selected constraint.

[0008] In some embodiments, (a)(iii) comprises on said interface (1) presenting a plurality of generation methods and a plurality of evaluation methods to said user and (2) receiving a selection of said at least one generation method and said at least one evaluation method from said user. In some embodiments, said at least one generation method generates at least one configuration from a search space and wherein said at least one generation method generates at least one allowed move within said search space. In some embodiments, said at least one evaluation method evaluates one or more objective functions and determines that at least zero constraints are satisfied for a given configuration.

[0009] In some embodiments, (a)(iv) comprises on said interface (1) presenting at least a subset of said plurality of algorithms to said user and (2) receiving a selection of said one or more algorithms from said user. In some embodiments, (a)(iv) comprises on said interface (1) presenting at least a subset of said plurality of computational platforms to said user and (2) receiving a selection of said one or more computational platforms from said user. In some embodiments, (b) comprising aiding or suggesting said selection by said interface.

[0010] In some embodiments, (b) comprises executing said one or more algorithms until a stopping criterion is met on said one or more computational platforms. In some embodiments, said interface is configured to communicate with said plurality of computational platforms over a network.

[0011] In some embodiments, said one or more optimization specifications further comprises at least one constraints handling procedure. In some embodiments, said at least one constraints handling procedure comprises at least one of: incorporating static and adaptive penalty terms into

said one or more objective function, rejection sampling or rejection of infeasible configurations, extending the search space to include Lagrange multipliers, or embedding techniques, optionally, wherein said embedding techniques comprise one or more of constraint programming, linear programming, and mixed-integer programming in said search space to restrict said search space. In some embodiments, said one or more algorithms employs said at least one constraints handling procedure.

[0012] In some embodiments, said one or more algorithms comprises one or more of: local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.

[0013] In some embodiments, said search space comprises one or more of: a search space comprising binary variables, a search space comprising integer variables, a search space comprising categorical variables, a search space comprising permutations, or a search space comprising continuous variables.

[0014] In some embodiments, said one or more computational platforms comprises one or more of: a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), central processing unit (CPU), graphics processing unit (GPU), a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a quantum annealer, integrated photonic coherent Ising machine, or an optical quantum computer.

[0015] In some embodiments, said optimization problem comprises one or more of: combinatorial optimization, constrained and unconstrained binary optimization, constrained and unconstrained integer optimization, constrained and unconstrained mixed-integer optimization, constrained and unconstrained continuous optimization, multi-objective optimization, or optimization under uncertainty.

[0016] In some embodiments, said one or more objective functions comprise one or more of: a linear objective function, a quadratic objective function, a non-linear objective function, a non-linear convex objective function, a non-linear non-convex objective function, a look-up table objective function, or a black-box objective function. In some embodiments, said at least zero constraints comprises one or more of: a linear equality constraint, a linear inequality constraint, a quadratic constraint, a non-linear parametric constraint, a look-up table constraint, or a black-box constraint.

[0017] In another aspect, a system for solving an optimization problem is provided. The system may comprise: an interface of a digital computer, said digital computer operatively coupled to a memory comprising instructions, wherein said digital computer is configured to execute said

instructions to at least: receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms, wherein said interface is in communication with a computational platform configured to execute said one or more algorithms, to yield a solution to said optimization problem, wherein said solution to said optimization problem from said computational platform is provided at said interface.

[0018] In some embodiments, said interface is a user interface (UI). In some embodiments, said UI is a graphical UI (GUI). In some embodiments, said one or more optimization problem specifications comprises at least one of: a search space comprising configurations of a plurality of search spaces, one or more objective functions of a plurality of objective functions, an allowed move of a plurality of allowed moves, or at least zero constraints of a plurality of constraints. In some embodiments, said interface is configured to (1) present at least a subset of said plurality of optimization problem specifications to said user and (2) receive a selection of said one or more optimization problem specifications from said user. In some embodiments, said interface is configured to receive a selection of said one or more optimization problem specifications from said user, wherein said selection comprises one or more of a selected search space comprising configurations, a selected allowed move, one or more selected objective functions, and at least one selected constraint.

[0019] In some embodiments, said interface is configured to (1) present a plurality of generation methods and a plurality of evaluation methods to said user and (2) receive a selection of said at least one generation method and said at least one evaluation method from said user. In some embodiments, said at least one generation method is configured to generate at least one configuration from a search space and wherein said at least one generation method generates at least one allowed move within said search space. In some embodiments, said at least one evaluation method is configured to evaluate one or more objective functions and determines that at least zero constraints are satisfied for a given configuration.

[0020] In some embodiments, said interface is configured to (1) present at least a subset of said plurality of algorithms to said user and (2) receive a selection of said one or more algorithms from said user. In some embodiments, said interface is configured to (1) present at least a subset of said plurality of computational platforms to said user and (2) receive a selection of said one or

more computational platforms from said user. In some embodiments, said selection is aided or suggested by said interface.

[0021] In some embodiments, said digital computer is configured to execute said instructions to execute said one or more algorithms until a stopping criterion is met on said one or more computational platforms. In some embodiments, said interface is configured to communicate with said plurality of computational platforms over a network.

[0022] In some embodiments, said one or more optimization specifications further comprises at least one constraints handling procedure. In some embodiments, said at least one constraints handling procedure comprises at least one of: incorporating static and adaptive penalty terms into the one or more objective functions, rejection sampling or rejection of infeasible configurations, or extending the search space to include Lagrange multipliers, or embedding techniques such as constraint programming, linear programming, and mixed-integer programming in the search space to restrict the search space. In some embodiments, said one or more algorithms employs said at least one constraints handling procedure.

[0023] In some embodiments, said one or more algorithms comprises one or more of: local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.

[0024] In some embodiments, said search space comprises one or more of: a search space comprising binary variables, a search space comprising integer variables, a search space comprising categorical variables, a search space comprising permutations, or a search space comprising continuous variables.

[0025] In some embodiments, said one or more computational platforms comprises one or more of: a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), central processing unit (CPU), graphics processing unit (GPU), a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a quantum annealer, integrated photonic coherent Ising machine, or an optical quantum computer.

[0026] In some embodiments, said optimization problem comprises one or more of: combinatorial optimization, constrained and unconstrained binary optimization, constrained and unconstrained integer optimization, constrained and unconstrained mixed-integer optimization, constrained and unconstrained continuous optimization, multi-objective optimization, or optimization under uncertainty.

[0027] In some embodiments, said one or more objective functions comprise one or more of: a linear objective function, a quadratic objective function, a non-linear objective function, a non-

linear convex objective function, a non-linear non-convex objective function, a look-up table objective function, or a black-box objective function.

[0028] In some embodiments, said at least zero constraints comprises one or more of: a linear equality constraint, a linear inequality constraint, a quadratic constraint, a non-linear parametric constraint, a look-up table constraint, or a black-box constraint.

[0029] In another aspect, a non-transitory computer readable medium comprising machine-executable code that upon execution by a digital computer in communication with a computational platform implements a method for solving an optimization problem from a user is provided. The method may comprise: (a) using an interface of a digital computer to receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms; (b) using said interface to transmit a request to said one or more computational platforms to execute said one or more algorithms to yield a solution to said optimization problem; and (c) providing said solution to said optimization problem from said computational platform at said interface.

INCORPORATION BY REFERENCE

[0030] All publications, patents, and patent applications mentioned in this specification are herein incorporated by reference to the same extent as if each individual publication, patent, or patent application was specifically and individually indicated to be incorporated by reference. To the extent publications and patents or patent applications incorporated by reference contradict the disclosure contained in the specification, the specification is intended to supersede and/or take precedence over any such contradictory material.

BRIEF DESCRIPTION OF THE DRAWINGS

[0031] The novel features of the invention are set forth with particularity in the appended claims. A better understanding of the features and advantages of the present invention will be obtained by reference to the following detailed description that sets forth illustrative embodiments, in which the principles of the invention are utilized, and the accompanying drawings (also "Figure" and "FIG." herein), of which:

[0032] FIG. 1 is a flowchart of a method for solving an optimization problem using a flexible modular approach, in accordance with some embodiments disclosed herein.

[0033] FIG. 2 is a flowchart of a method for solving an optimization problem using the flexible modular computational framework FOCUS, in accordance with some embodiments disclosed herein.

[0034] FIG.3 is a flowchart of a procedure for providing an indication of the optimization problem, in accordance with some embodiments disclosed herein.

[0035] FIG.4 is a flowchart of a procedure for initializing a generation method, in accordance with some embodiments disclosed herein.

[0036] FIG.5 is a flowchart of a procedure for initializing an evaluation method, in accordance with some embodiments disclosed herein.

[0037] FIG.6 is a flowchart of a procedure for integrating a generation method, an evaluation method, and a constraints handling procedure into an interface, in accordance with some embodiments disclosed herein.

[0038] FIG. 7 is a diagram of a system for solving an optimization problem using a flexible modular approach, in accordance with some embodiments disclosed herein.

DETAILED DESCRIPTION

[0039] While various embodiments of the invention have been shown and described herein, it will be obvious to those skilled in the art that such embodiments are provided by way of example only. Numerous variations, changes, and substitutions may occur to those skilled in the art without departing from the invention. It should be understood that various alternatives to the embodiments of the invention described herein may be employed.

[0040] Unless otherwise defined, all technical terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this invention belongs. As used in this specification and the appended claims, the singular forms "a," "an," and "the" include plural references unless the context clearly dictates otherwise. Any reference to "or" herein is intended to encompass "and/or" unless otherwise stated.

[0041] Whenever the term "at least," "greater than," or "greater than or equal to" precedes the first numerical value in a series of two or more numerical values, the term "at least," "greater than" or "greater than or equal to" applies to each of the numerical values in that series of numerical values. For example, greater than or equal to 1, 2, or 3 is equivalent to greater than or equal to 1, greater than or equal to 2, or greater than or equal to 3.

[0042] Whenever the term "no more than," "less than," or "less than or equal to" precedes the first numerical value in a series of two or more numerical values, the term "no more than," "less than," or "less than or equal to" applies to each of the numerical values in that series of numerical

values. For example, less than or equal to 3, 2, or 1 is equivalent to less than or equal to 3, less than or equal to 2, or less than or equal to 1.

[0043] In the following detailed description, reference is made to the accompanying figures, which form a part hereof. In the figures, similar symbols typically identify similar components, unless context dictates otherwise. The illustrative embodiments described in the detailed description, figures, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing from the scope of the subject matter presented herein. It will be readily understood that the aspects of the present disclosure, as generally described herein, and illustrated in the figures, can be arranged, substituted, combined, separated, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

[0044] An optimization problem or a general optimization problem may be a problem of increasing or decreasing one or more of a set of objective functions until a stopping criterion is satisfied. A stopping criterion may be a threshold condition. In some cases, the optimization problem may comprise finding a maximum or near maximum. In some cases, the optimization problem may comprise find a minimum or near minimum. In some cases, the optimization problem may comprise finding a configuration or configurations of one or more decision variables among a set of possible configurations D to increase or decrease one or more of a set of objective functions f(x) to a threshold condition. In some cases, the optimization problem may comprise increasing or decreasing one or more of a set of objective functions such that one or more of a set of constraints C(x) is satisfied. An optimization problem may be denoted by P herein. Each constraint may represent a relationship among decision variables x. In a general optimization problem, there may be no restriction on decision variables' type and value. In a general optimization problem, there may be no restriction on the form of the objective functions and constraints. In some cases, the definition may include finding a near optimal solution or a number of near optimal solutions.

[0045] An objective function may be a function of decision variables which may be increased or decreased by varying input values and computing the value of the function. In some cases, an objective function may be a real function. In some cases, an objective function may be a function of decision variables which may be increased or decreased by choosing input values from within a set of input values and computing the value of the function. In some cases, the objective function may be a parametric function or a black-box function of decision variables. The objective function may accept a given configuration as an input and output a real value.

[0046] A configuration or solution candidate may comprise one or more of a set of possible values of decision variables. In some cases, a configuration may be a member of a set of possible

configurations and may represent one configuration of decision variables. A configuration may be a member of the set of possible configurations, D, and may represent one configuration of variables, x.

[0047] A feasible configuration may be a configuration which satisfies each of the set of constraints. In some cases, a feasible configuration may be a configuration x which satisfies problem constraints in the set C(x).

[0048] An infeasible configuration may be a configuration which does not satisfy at least one constraint of the set of constrains. In some cases, an infeasible configuration may be a configuration x which does not satisfy at least one of the constraints in the set C(x).

[0049] A search space or a solution space may comprise a set of possible configurations. In some cases, a search space refers to a set of all possible configurations. In some cases, a search space refers to a set of searched configurations. In some cases, a search space is the set of possible configurations D. In some applications, the search space may be further adjusted by directly incorporating all, at least some, or at least one of the problem constraints such that it may include feasible configurations subject to all, at least some, or at least one of the problem constraints.

[0050] A move or allowed move may be a change from a first configuration to a second configuration. A first configuration and a second configuration may be within a search space. A first configuration and a second configuration may be within of a set of possible configurations. In some cases, a move may be generated subject to a prescription describing how the values of a given configuration change to generate a new configuration. In some cases, a move may change a configuration x to a second configuration x'.

[0051] A generation method may comprise one or more procedures related to generating or changing a configuration. In some cases, a generation method comprises one or more procedures that describe how to generate a configuration and a move. In some cases, a generation method comprises one or more procedures that describe how to transform a given configuration into a new configuration using a move.

[0052] An evaluation method may comprise one or more procedures related to evaluating at least one objective function at a configuration. An evaluation method may comprise one or more procedures that evaluate at least one objective function and at least zero constraints for a given configuration and a move.

[0053] A constraint may comprise a condition of an optimization problem which a solution must satisfy. Constraints may be integrated into the systems and methods herein using constraints handling procedures. A constraints handling procedure may guide the search toward feasible configurations. The constraints handling procedure may be embedded in the interface. A

constraints handling procedure may be useful in the case wherein the search space includes both feasible and infeasible configurations and the generation method does not necessarily draw feasible configurations from the search space or does not generate moves that yield new feasible configurations when applied on the current feasible/infeasible configuration.

[0054] An optimization algorithm may comprise a set of operations that describe how to find a set of optimal or near optimal configurations for the optimization problem *P*.

[0055] An interface may comprise a computer implemented connection between the various components (e.g. software to software, software to hardware, software to user, etc.) of the system for solving an optimization problem. The interface may comprise a flexible modular configuration. For example, the interface may provide access to one or more modules. The interface may do one or more of the following: facilitate access to methods to draw configurations/moves, evaluate a set of objective functions and constraints for a given configuration/move, address the constraints when acting as an abstraction layer between algorithms and optimization problems, etc.

[0056] The interface may be a modular interface. In some cases, an interface may be a modular interface which connects aspects of the code represented as interchangeable modules. In some cases, the interface may provide access to one or more modules of a method for solving an optimization problem. In some cases, one or more modules may be provided which allows a user to provide one or more indications of a method for solving an optimization problem. An indication may comprise a selection from available options or a user may provide his or her own machine readable code.

[0057] For example, a module for selecting optimization problem specifications may comprise, on the interface, presenting at least a subset of a plurality of optimization problem specifications to a user and receiving a selection of one or more optimization problem specifications from the user. A selection of a subset of a plurality of optimization problem specifications may comprise one or more of a selected search space comprising configurations, a selected allowed move, one or more selected objective functions, and at least one selected constraint. In some cases, a user may provide his or her own machine-readable code comprising one or more optimization problem specifications.

[0058] For example, a module for selecting a generation method comprises, on said interface, presenting a plurality of generation methods to said user and receiving a selection of said at least one generation method from said user. In some cases, the at least one generation method generates at least one configuration from a search space. In some cases, the at least one generation method generates at least one allowed move within said search space. In some cases, a user may provide his or her own machine-readable code comprising a generation method.

[0059] For example, a module for selecting an evaluation method comprises, on said interface, presenting a plurality of evaluation methods to said user and receiving a selection of said at least one evaluation method from said user. In some cases, the at least one evaluation method evaluates one or more objective functions. In some cases, the at least one evaluation method determines that zero constraints are satisfied for a given configuration. In some cases, a user may provide his or her own machine-readable code comprising an evaluation method.

[0060] If a generation method and an evaluation method are provided, the interface may wrap the generation and the evaluation methods to draw and evaluate configurations/moves. Wrapping the generation and evaluation methods by the interface may be modular and flexible, but it may have some overhead. In some cases, a generation method and an evaluation method may be wrapped together to form a module. In some cases, for better performance, the user may decide to directly incorporate the methods for generation and evaluation of configurations and moves into the interface. For example, a user may incorporate their own methods by providing or writing machine readable code configured to: draw configurations/moves, to evaluate a set of objective functions and constraints for a given configuration/move, to deal with the constraints when acting as an abstraction layer between algorithms and optimization problems, etc.

[0061] For example, a module for selecting an algorithm comprises, on the interface, presenting at least a subset of the plurality of algorithms to the user and receiving a selection of the one or more algorithms from the user. In some cases, a user may provide his or her own machine-readable code comprising an optimization algorithm.

[0062] For example, a module for selecting a computational platform comprises, on the interface, presenting at least a subset of the plurality of computational platforms to the user and receiving a selection of the one or more computation platforms from the user. In some cases, the computation platform may be aided or suggested by the interface.

[0063] Because each module may comprise various implementations of the code, aspects of the present disclosure may advantageously provide a flexible platform for solving optimization problems. Because each module may allow a user to provide their own problem indications, the platform may provide additional flexibility to adapt to sophisticated users while providing more specific options to less sophisticated users.

[0064] As another non-limiting advantage, the interface may act as an abstraction layer between the optimization problem and the algorithms used to solve the optimization problem. In some cases, the interface may shield a user from machine code. For example, in some cases, the interface may aid or select various indications of various portions of an optimization problem, for example: the optimization problem, the one or more optimization problem specifications of a plurality of optimization problem specifications, the at least one generation method and at least

one evaluation method, the one or more algorithms of a plurality of algorithms, and a selection of one or more computational platforms of a plurality of computational platforms. In some cases, the various indications may be provided automatically by the interface, e.g., without the input of a user. Such cases may include pre-implemented interfaces or pre-implemented generation methods as disclosed elsewhere herein. In some cases, the various options for indications may be automatically removed by the interface, e.g., without the input of a user, if such options are non-feasible for the particular optimization problem.

[0065] As another non-limiting advantage, the flexible nature of the interface may facilitate implementation of various types of optimization problems on various hardware types including classical and non-classical computational platforms. The interface may facilitate implementation of various types of optimization problems on various hardware types including binary solvers. In some cases, an interface as described herein may be agnostic to the computational platform. In some cases, an interface as described herein may facilitate accesses to various types of computational platforms. In some cases, at least one of these types of computational platforms is a non-classical computer, e.g., a quantum computer. The methods included in the interface may address constraints, if any.

[0066] In some cases, a user interface may be provided in addition to the modular interface. For example, the user interface may allow a user to provide one or more indications of various portions of an optimization problem including, for example: the optimization problem, the one or more optimization problem specifications of a plurality of optimization problem specifications, the at least one generation method and at least one evaluation method, the one or more algorithms of a plurality of algorithms, and a selection of one or more computational platforms of a plurality of computational platforms. In some cases, the user interface may be a graphical user interface.

[0067] In some cases, a module for selecting a computational platform comprises or is operably connected to a hardware interface. The hardware interface may facilitate connection to various hardware types, including classical and non-classical computational platforms, to facilitate implementation of various types of optimization problems. The hardware interface may be configured to communicate with the plurality of computational platforms over a network. In some cases, the network is a local network. In some cases, the computational platform is remote to the modular interface. For example, a user interface may be on machine local to a user, a modular interface may be implemented via a networked server which is local or remote to a user, and a computational platform may be local or remote to each of the networked server or the user.

[0068] As another non-limiting advantage, the flexible nature of the interface may allow for implementation of the computationally expensive procedures in generation, evaluation, and/or

interface on special-purpose hardware if possible or on massively parallel devices such as an FPGA, GPU, and/or ASIC.

Optimization Problems

[0069] At least two classical optimization problems are used herein to illustrate the examples disclosed in FIGS. 1-6, e.g., a travelling salesperson problem and a vehicle routing problem. The travelling salesperson problem and the vehicle routing problem are shown as non-limiting examples meant to illustrate the functionality of the methods and systems disclosed herein. Other types of optimization problems are possible. Methods and systems disclosed herein may not be limited to solving optimization problems of a particular form. For example, methods and systems of the present disclosure may employ optimization problems in at least the following types, e.g., linear programming, mixed integer programming, QUBO, etc.

[0070] In some cases, an optimization problem comprises a travelling salesperson problem. The travelling salesperson problem may be formulated as follows: given a number of cities and distances between each city pair, find a shortest path to start from the depot, visit each city once, and return to the depot. A shortest path may be a shortest path found in a search of configuration up to a threshold condition. A threshold condition may comprise, for example, reaching a threshold number of iterations, a value being reduced below a threshold hold amount per iteration, etc.

[0071] In some cases, an optimization problem comprises a vehicle routing problem. The vehicle routing problem may be formulated as follows: given a number of cities and a number of vehicles, find assignment of cities to vehicles and a route for each vehicle such that the total distance travelled is minimized or reduced below a threshold condition. A threshold condition may comprise, for example, reaching a threshold number of iterations, a value being reduced below a threshold hold amount per iteration, etc.

Optimization algorithms

[0072] Examples of optimization algorithms include but are not limited to: greedy search, simulated annealing, rejection-free simulated annealing, tabu search, path relinking, parallel tempering, population annealing, constrained simulated annealing, simulated quantum annealing, genetic/evolutionary algorithm, constrained population annealing, constrained parallel tempering, exhaustive search, and other meta heuristic approaches.

[0073] In some cases, an optimization algorithm comprises a greedy algorithm. A greedy algorithm is a local search algorithm that iteratively improves the current configuration. It may initialize at a random configuration and its corresponding objective function values. At each iteration, it may construct a neighborhood around the current configuration. The neighborhood

may comprise a set of prospective moves. It may continue the search by accepting the move that improves the current objective function values the most, if any. It may update the current configuration to the new configuration generated after applying the selected move. This procedure may continue until reaching the number of iterations. In some cases, the number of iterations may be specified in by a user, provided based on estimated computing resources, etc.

[0074] In some cases, an optimization algorithm comprises simulated annealing. Simulated annealing (SA) is an iterative, generic heuristic method that emulates the process of physical annealing to find an optimum or near optimum of an optimization problem. SA is a discrete time Markov chain. SA may act on a non-increasing temperature schedule. It may initialize at a random configuration, and at each temperature it may prescribe a random move. The probability of transitioning to a second configuration, generated by applying the move to the current configuration, may be calculated according to a Metropolis-Hastings criterion. A Metropolis-Hastings criterion may promote diversification at high temperatures and intensification at low temperatures. As with many of heuristic algorithms, reaching a local optimum or near optimum at the lowest temperature is a likely scenario in SA. To increase the likelihood of reaching the global optimum or near optimum, this process may be repeated multiple times, each time starting from a random initial solution. [Ref: Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220 , 671-680.], which is incorporated entirely herein by reference for all purposes.

[0075] In some cases, an optimization algorithm comprises rejection-free simulated annealing. The rejection-free simulated annealing is a Markov chain algorithm that differs from the simulated annealing algorithm in at least two aspects. At each temperature, 1) it may propose a number of moves for the given current configuration, and 2) it may transition to a second configuration with probability one. The second configuration may be chosen according to a probability distribution, which probability distribution may be normalized according to the Boltzmann factors associated with each proposed move. [Ref: J. W. Greene and K. J. Supowit. Simulated annealing without rejected moves. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 5(1):221-228, 1986.], which is incorporated entirely herein by reference for all purposes.

[0076] In some cases, an optimization algorithm comprises a tabu search. Tabu search is an iterative improvement algorithm that maintains the search history by recording the moves chosen in the tabu list for a given number of iterations e.g., tabu tenure. This method may avoid cycling where the same region of the search space is repeatedly visited. The algorithm may begin at a random initial configuration. At each iteration, it may construct a number of moves for the current configuration, may accept the best non-tabu move, and may update the current

configuration to a second configuration. It may then proceed to the next iteration by adding the selected move to the tabu list. [Ref: Glover F., and Laguna M., Tabu Search, *In: Du DZ., Pardalos P.M. (eds) Handbook of Combinatorial Optimization*, Springer, Boston, MA (1998)], which is incorporated entirely herein by reference for all purposes.

[0077] In some cases, an optimization algorithm comprises path-relinking. Path relinking algorithm uses an internal local search solver such as simulated annealing or tabu search to first generate a selected solution list composed of a number of local optima or near optima. It may start from distinct configurations connecting pairs of local optima or near optima and may improve them using an internal local search solver. [Ref: Resendel M. G., and Ribeiro C. C., GRASP with Path-Relinking: Recent Advances and Applications, *In: Ibaraki T., Nonobe K., Yagiura M. (eds) Metaheuristics: Progress as Real Problem Solvers. Operations Research/Computer Science Interfaces Series*, Springer, Boston, MA (2005)], which is incorporated entirely herein by reference for all purposes.

[0078] In some cases, an optimization algorithm comprises parallel tempering. The parallel tempering Monte Carlo algorithm simulates M replicas of the original system at different temperatures. The parallel tempering may use periodic exchanges based on a Metropolis-Hastings criterion between neighboring temperatures. Replica-exchange moves may allow replicas to perform a random walk in temperature space. This may enable the algorithm to efficiently overcome energy barriers. Replicas at high temperatures may improve algorithmic mixing, whereas replicas at low temperatures may reach equilibrium on a shorter time scale compared to some simulations at a fixed low temperature. In some cases, the parallel tempering algorithm may be more efficient than the simulated annealing algorithm in optimization and sampling applications. [Ref: Hukushima K., and Nemoto K., Exchange Monte Carlo Method and Application to Spin Glass Simulations, *Journal of the Physical Society of Japan*, 65, 1604 (1996)], which is incorporated entirely herein by reference for all purposes.

[0079] In some cases, an optimization algorithm comprises population annealing. Population annealing is a sequential Monte Carlo algorithm that performs a sequence of resampling and Monte Carlo simulation operations. It initializes with a population of configurations generated at random. At each temperature, it first resamples the population at the temperature such that the population at the current temperature has a Boltzmann distribution. It then proceeds by performing a number of Monte Carlo operations for each configuration in the resampled population. The resampling/simulation operations continue for a given number of times. [Ref: Wang W., Machta J., and Katzgraber H. G., Population Annealing: Theory and Application in Spin Glasses, *Physics Review E*, 92, 961 (2015)], which is incorporated entirely herein by reference for all purposes.

[0080] In some cases, an optimization algorithm comprises constrained simulated annealing. Constrained simulated annealing performs a sequence of Monte Carlo simulation operations. The sequence of Monte Carlo simulation operations may be performed in an extended search space combining decision variables and Lagrange multipliers. For a minimization/maximization problem, the search may represent a Markov Chain that performs probabilistic descents/ascent in the original solution space and probabilistic ascents/descents in the Lagrange multipliers space. [Ref: Wah B. W., Wang T., Simulated Annealing with Asymptotic Convergence for Nonlinear Constrained Global Optimization, *In: Jaffar J. (eds) Principles and Practice of Constraint Programming – Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg (1999)], which is incorporated entirely herein by reference for all purposes.

[0081] In some cases, an optimization algorithm comprises a genetic/evolutionary algorithm. Evolutionary algorithms are population-based improvement algorithms that are inspired by evolutionary theory. Although there are various ways to implement evolutionary mechanisms, the general idea is to improve the average fitness of the population through exposing each individual in the population to external pressure that leads to the survival of the fittest individuals. [Ref: Andrew N. Sloss and Steven Gustafson, 2019 Evolutionary Algorithms Review, arXiv:1906.08870, 2019], which is incorporated entirely herein by reference for all purposes.

Objective functions

[0082] Each objective function in the set of objective functions may represent one or more of a parametric, non-parametric, or black-box real-valued function of the decision variables that may be reduced or increased to a threshold condition. The input and output to each objective function may be, in respective terms, a configuration from the set of possible configurations and a real value. Some examples of minimizing an objective function include the total time and distance traveled in routing applications, the total time to process jobs in scheduling applications, the number of resources used in resource planning applications, the operational cost in healthcare and service applications, etc. Some examples of maximizing an objective function include the profit of investment in finance applications, the reliability in asset management applications, and the similarity in clustering applications.

Computational platform

[0083] Computational platform may comprise various types of hardware in any combination. Each of the hardware may be used as part of the system, to execute the whole method, or any part of it, alone, or in combination with other hardware. In some cases, the hardware may be used for

various operations of the methods disclosed herein, including, for example, one or more of the following:

- Evaluation of the objective functions given an initial configuration.
- Evaluation of the constraints given an initial configuration.
- Evaluation of the objective functions given a configuration and objective functions' values, and a prospective move.
- Evaluation of the constraints given a configuration and objective functions' values, and a prospective move.
- Execution of an optimization algorithm or a part of an optimization algorithm.
- Generation of random numbers.
- Calculation of Boltzmann factors for prospective moves.
- Generation of new moves.
- Generation of random solutions.
- Execution of functions of the interface, including a part or all of the above.

[0084] A computational platform may comprise a central processing unit (CPU). A CPU may be a low latency integrated circuit chip which comprises the main processor in a computer. A CPU may execute instructions as given by an algorithm. A CPU may comprise a component configured to do one or more of the following: executing arithmetic and logic operations, registering that store the results of those operations, and directing the operations of the former using a control unit.

[0085] A computational platform may comprise a graphics processing unit (GPU). A GPU may be specialized electronic circuit optimized for high throughput - can perform the same set of operations in parallel on many data blocks at a time.

[0086] A computational platform may comprise a field-programmable gate array (FPGA). An FPGA may comprise an integrated circuit chip that comprises configurable logic blocks and programmable interconnects. Can be programmed after manufacturing to execute custom algorithms.

[0087] A computational platform may comprise an application-specific integrated circuit (ASIC). An ASIC may be an integrated circuit chip that is customized to run a specific algorithm and cannot be programmed after manufacturing.

[0088] A computational platform may comprise a tensor processing unit (TPU). A TPU may comprise a proprietary type of ASIC developed for low bit precision processing by Google Inc., see Patent Application US 2016/0342891A1, which is incorporated entirely herein by reference for all purposes.

[0089] A computational platform may comprise a tensor streaming processor (TSP). A TSP may be a domain-specific programmable integrated chip that is designed for linear algebra computations as they may be performed in Artificial Intelligence applications [Ref: https://groq.com/wp-content/uploads/2020/01/Groq-Rocks-NNs-Linley-Group-MPR-2020Jan06.pdf], which is incorporated entirely herein by reference for all purposes

Non-classical computer

[0090] The present disclosure provides systems and methods that may include quantum computing or use of quantum computing. Quantum computers may be able to solve certain classes of computational tasks more efficiently than classical computers. However, quantum computation resources may be rare and expensive, and may involve a certain level of expertise to be used efficiently or effectively (e.g., cost-efficiently, or cost-effectively). A number of parameters may be tuned in order for a quantum computer to deliver its potential computational power.

[0091] Quantum computers (or other types of non-classical computers) may be able to work alongside classical computers as co-processors. A hybrid architecture (e.g., computing system) comprising a classical computer and a quantum computer can be very efficient for addressing complex computational tasks, such as quantum chemistry simulations. Systems and methods disclosed herein may be able to efficiently and accurately implement a quantum problem on a non-classical computer with a reduced number of single and two-qubit gates.

[0092] Although the present disclosure has referred to quantum computers, methods and systems of the present disclosure may be employed for use with other types of computers, which may be non-classical computers. Such non-classical computers may comprise quantum computers, hybrid quantum computers, quantum-type computers, or other computers that are not classical computers. Examples of non-classical computers may include, but are not limited to, Hitachi Ising solvers, coherent Ising machines based on optical parameters, and other solvers which utilize different physical phenomena to obtain more efficiency in solving particular classes of problems.

[0093] In some cases, a quantum computer may comprise one or more adiabatic quantum computers, quantum gate arrays, one-way quantum computers, topological quantum computers, quantum Turing machines, superconductor-based quantum computers, trapped ion quantum computers, trapped atom quantum computers, optical lattices, quantum dot computers, spin-based quantum computers, spatial-based quantum computers, Loss-DiVincenzo quantum computers, nuclear magnetic resonance (NMR) based quantum computers, solution-state NMR quantum computers, solid-state NMR quantum computers, solid-state NMR Kane quantum computers, electrons-on-helium quantum computers, cavity-quantum-electrodynamics based quantum computers, molecular magnet

quantum computers, fullerene-based quantum computers, linear optical quantum computers, diamond-based quantum computers, nitrogen vacancy (NV) diamond-based quantum computers, Bose–Einstein condensate-based quantum computers, transistor-based quantum computers, and rare-earth-metal-ion-doped inorganic crystal based quantum computers. A quantum computer may comprise one or more of: quantum annealers, Ising solvers, optical parametric oscillators (OPO), and gate models of quantum computing.

[0094] A computational platform may comprise a non-classical computer which is a gate model quantum computer. A gate model quantum computer may comprise quantum bits, referred to as qubits, and a network of quantum logic gates. In a gate model quantum computer computations may be performed by initializing quantum states, running them through a sequence of quantum gates (a quantum circuit) and performing measurements on the states.

[0095] A computational platform may comprise a non-classical computer which is a quantum annealer. A quantum annealer may comprise a quantum mechanical system consisting of manufactured qubits, local field biases, and couplings between pairs of qubits. The biases and couplings may be controllable. A quantum annealer may allow the quantum annealer to be programmed such that it can be used as a heuristic solver for Ising problems, equivalent to quadratic unconstrained binary optimization (QUBO) problems. See McGeoch, Catherine C. and Cong Wang, (2013), "Experimental Evaluation of an Adiabatic Quantum System for Combinatorial Optimization", Computing Frontiers, May 14 16, 2013 and the Patent Application US 2006/0225165, which is incorporated entirely herein by reference for all purposes.

[0096] A computational platform may comprise a non-classical computer which is an optical computing device. Another example of an analogue system suitable for technology disclosed herein is an optical device. In some cases, the optical device comprises a network of optical parametric oscillators (OPOs) as disclosed in the patent applications US20160162798 and WO2015006494 A1, which are each incorporated herein by reference in their entireties.

[0097] A computational platform may comprise a non-classical computer which is an integrated photonic coherent ising machine. An example of an analogue system suitable for technology disclosed herein is an Integrated photonic coherent Ising machine disclosed in the patent application US2018/0267937A1, which is incorporated entirely herein by reference for all purposes. In some cases, an Integrated photonic coherent Ising machine may comprise a combination of nodes and a connection network solving a particular Ising problem. In some cases, the combination of nodes and the connection network may form an optical computer that is adiabatic. The combination of the nodes and the connection network may non-deterministically solve an Ising problem when the values stored in the nodes reach a steady state to minimize the energy of the nodes and the connection network. Values stored in the nodes at the minimum

energy level may be associated with values that solve a particular Ising problem. The stochastic solutions may be used as samples from the Boltzmann distribution defined by the Hamiltonian corresponding to the Ising problem.

Classical Computer

[0098] In some cases, the systems, media, networks, and methods described herein comprise a classical computer, or use of the same. In some cases, a classical computer may comprise a digital computer. In some cases, the classical computer includes one or more hardware central processing units (CPUs) that carry out the classical computer's functions. In some cases, the classical computer further comprises an operating system (OS) configured to perform executable instructions. In some cases, the classical computer is connected to a computer network. In some cases, the classical computer is connected to the Internet such that it accesses the World Wide Web. In some cases, the classical computer is connected to a cloud computing infrastructure. In some cases, the classical computer is connected to an intranet. In some cases, the classical computer is connected to a data storage device.

[0099] In accordance with the description herein, suitable classical computers may include, by way of non-limiting examples, server computers, desktop computers, laptop computers, notebook computers, sub-notebook computers, netbook computers, netpad computers, set-top computers, media streaming devices, handheld computers, Internet appliances, mobile smartphones, tablet computers, personal digital assistants, video game consoles, and vehicles. Smartphones may be suitable for use with methods and systems described herein. Select televisions, video players, and digital music players, in some cases with computer network connectivity, may be suitable for use in the systems and methods described herein. Suitable tablet computers may include those with booklet, slate, and convertible configurations.

[0100] In some cases, the classical computer includes an operating system configured to perform executable instructions. The operating system may be, for example, software, including programs and data, which manages the device's hardware and provides services for execution of applications. Suitable server operating systems include, by way of non-limiting examples, FreeBSD, OpenBSD, NetBSD®, Linux, Apple® Mac OS X Server®, Oracle® Solaris®, Windows Server®, and Novell® NetWare®. Suitable personal computer operating systems may include, by way of non-limiting examples, Microsoft® Windows®, Apple® Mac OS X®, UNIX®, and UNIX-like operating systems such as GNU/Linux®. In some cases, the operating system is provided by cloud computing. Suitable mobile smart phone operating systems may include, by way of non-limiting examples, Nokia® Symbian® OS, Apple® iOS®, Research In Motion® BlackBerry OS®, Google® Android®, Microsoft® Windows Phone® OS, Microsoft® Windows Mobile® OS, Linux®, and Palm® WebOS®. Suitable media streaming device operating systems may include,

by way of non-limiting examples, Apple TV®, Roku®, Boxee®, Google TV®, Google Chromecast®, Amazon Fire®, and Samsung® HomeSync®. Suitable video game console operating systems may include, by way of non-limiting examples, Sony® PS3®, Sony® PS4®, Microsoft® Xbox 360®, Microsoft Xbox One, Nintendo® Wii®, Nintendo® Wii U®, and Ouya®.

[0101] In some cases, the classical computer includes a storage and/or memory device. In some cases, the storage and/or memory device is one or more physical apparatuses used to store data or programs on a temporary or permanent basis. In some cases, the device is volatile memory and requires power to maintain stored information. In some cases, the device is non-volatile memory and retains stored information when the classical computer is not powered. In some cases, the non-volatile memory comprises flash memory. In some cases, the non-volatile memory comprises dynamic random-access memory (DRAM). In some cases, the non-volatile memory comprises ferroelectric random access memory (FRAM). In some cases, the non-volatile memory comprises phase-change random access memory (PRAM). In other embodiments, the device is a storage device including, by way of non-limiting examples, CD-ROMs, DVDs, flash memory devices, magnetic disk drives, magnetic tapes drives, optical disk drives, and cloud computing based storage. In some cases, the storage and/or memory device is a combination of devices such as those disclosed herein.

[0102] In some cases, the classical computer includes a display to send visual information to a user. In some cases, the display is a cathode ray tube (CRT). In some cases, the display is a liquid crystal display (LCD). In some cases, the display is a thin film transistor liquid crystal display (TFT-LCD). In some cases, the display is an organic light emitting diode (OLED) display. In some cases, on OLED display is a passive-matrix OLED (PMOLED) or active-matrix OLED (AMOLED) display. In some cases, the display is a plasma display. In other embodiments, the display is a video projector. In some cases, the display is a combination of devices such as those disclosed herein.

[0103] In some cases, the classical computer includes an input device to receive information from a user. In some cases, the input device is a keyboard. In some cases, the input device is a pointing device including, by way of non-limiting examples, a mouse, trackball, track pad, joystick, game controller, or stylus. In some cases, the input device is a touch screen or a multi-touch screen. In some cases, the input device is a microphone to capture voice or other sound input. In some cases, the input device is a video camera or other sensor to capture motion or visual input. In some cases, the input device is a Kinect, Leap Motion, or the like. In some cases, the input device is a combination of devices such as those disclosed herein.

[0104] The methods and systems disclosed herein may provide a novel flexible and modular computational framework for solving constrained optimization problems.

[0105] The methods and systems disclosed herein may not restrict the solution space of an optimization problem into a prescribed space, such as an unconstrained binary space when using the quantum annealer. This may allow user to benefit from the new computational technologies without converting to a prescribed space.

[0106] Methods and systems disclosed herein may provide a flexible modular approach, which approach may allow for incorporation of problem-specific knowledge in the generation and evaluation methods, such that the algorithms may not need to be changed for every problem.

[0107] Methods and the systems disclosed herein may encapsulate the strategy of the optimization methods in three high-level methods: generation, evaluation, and interface methods. The generation method may provide procedures to generate solution candidates and to walk in the search space by going from one solution candidate to another solution. The evaluation method comprises procedures to evaluate the objective functions, which may also comprise constraints for a given solution candidate. The generation and evaluation methods may be abstracted away from various optimization algorithms in the interface.

[0108] Methods and systems disclosed herein may not be limited to solving optimization problems of a particular form, e.g. linear programming, mixed integer programming, QUBO, etc. The interface may be flexible. For example, the interface may provide various options for a user to build an optimization problem. In some cases, there may be no restriction on how the procedures in the generation and evaluation methods are defined. If an option is missing, a user can provide code which can be used to adapt methods and systems disclosed herein. Methods and systems disclosed herein can handle a large number of optimization problems with various parametric and non-parametric functional forms representing the objective functions and constraints.

[0109] In some cases, the methods and systems disclosed herein may be equipped with pre-implemented generation methods for search spaces such as binary, permutation, categorical, and continuous. Multiple existing generation methods might be combined to create a new generation method for mixed solution spaces. The generation methods can be further adjusted to benefit from the knowledge and the structure specific to the given optimization problem. For a custom solution space, a new generation method can be implemented. In some cases, the methods and systems disclosed herein may, given a generation method and an evaluation method for an optimization problem, switch among various pre-implemented algorithms with no extra adjustment.

[0110] In some cases, the methods and systems disclosed herein includes pre-implemented interfaces that handle problem constraints utilizing various approaches. In some cases, it may be possible to use several different interfaces for solving a given optimization problem. The

performance of each interface may be changed based on the problem characteristics. Custom interfaces can be added for better performance.

[0111] In some cases, the methods and systems disclosed herein may comprise a flexible modular design, which allows for utilization of new technologies in different methods where the computationally expensive procedures in generation, evaluation, and/or interface are implemented on special-purpose hardware if possible or on massively parallel devices such as FPGA, GPU, and/or ASIC.

[0112] FIG. 1 is a flowchart of a method for solving an optimization problem using a flexible modular approach, in accordance with some embodiments disclosed herein.

[0113] According to processing operation 100, an indication of an optimization problem is obtained as an input request using a communications interface. In some cases, the optimization problem may comprise any optimization problem described herein or any combination thereof. In some cases, the optimization problem is a combinatorial optimization problem. In some cases, the optimization problem is a constrained or unconstrained binary optimization problem. In some cases, the optimization problem is a constrained or unconstrained integer optimization problem. In some cases, the optimization problem is a constrained or unconstrained mixed-integer optimization problem. In some cases, the optimization problem is a constrained or unconstrained continuous optimization problem. In some cases, the optimization problem is a multi-objective optimization problem or an optimization problem under uncertainty.

[0114] Still referring to FIG. 1 and according to processing operation 102, data representative of a search space comprising configurations, of at least one objective function and of at least zero constraints is provided using the optimization problem. In some cases, the data representative is provided by a user. In some cases, the data representative is extracted by the computer-implemented method. In some cases, the search space may be of various types. In particular, the search space may be any search space described herein or any combination thereof. In some cases, the search space comprises binary variables. In some cases, the search space comprises integer variables. In some cases, the search space comprises categorical variables. In some cases, the search space comprises permutations. In some cases, the search space comprises continuous variables. In some cases, the search space is further adjusted by directly incorporating some of the problem constraints such that it may include feasible configurations.

[0115] According to processing operation 104, an indication of at least one method for generating at least one configuration and at least one allowed move is provided. In some cases, the generation method may be provided by a user. In some cases, the generation method is provided by the computer-implemented method. In some cases, the generation method may

comprise procedures that describe how to generate a configuration and a move, and how to transform a given configuration into a new configuration using a move.

[0116] According to processing operation 106, an indication of at least one method for evaluating the objective functions and the constraints for a given configuration is provided. In some cases, the evaluation method may be provided by a user. In some cases, the evaluation method is provided by the computer-implemented method. In some cases, the evaluation method may comprise procedures that evaluate at least one objective function and at least zero constraints for a given configuration and a move.

[0117] Still referring to FIG. 1 and according to processing operation 108, at least one algorithm is selected. In some cases, the algorithm may be selected by a user, or by a computer-implemented method. The selected algorithm utilizes the methods as mentioned above to generate the configurations and the allowed moves and to evaluate the corresponding set of objective functions and constraints for a given configuration. In some cases, the algorithm may be of various types, it may be any algorithm, such as any algorithm described herein. In some cases, the algorithm is a member of the group consisting of local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.

[0118] According to processing operation 110, a computational platform comprising at least one processor is selected. In some cases, the computational platform may be of various types. The computational platform may be any suitable computational platform such as any computational platform described herein with respect to the system shown in FIG. 7. In some cases, the computational platform comprises at least one member of the group consisting of a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), central processing unit (CPU), graphics processing unit (GPU), a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a quantum annealer and an optical quantum computer.

[0119] Still referring to FIG. 1 and according to processing operation 112, each of the at least one algorithm is executed on the computational platform until a stopping criterion is met. In some cases, the stopping criterion may be of various types for each of the selected algorithms. In some cases, the stopping criterion is met when the number of generated configurations reaches a certain value. In some cases, the stopping criterion is met when the convergence is established. In some cases, the algorithms may be executed in parallel. In some cases, the algorithms are

executed sequentially. In some cases, the results of one of the algorithms is fed into another algorithm.

[0120] According to processing operation 114, results comprising at least one solution to the optimization problem are provided. In some cases, the result may be provided in various ways. In some cases, the results are provided via the communications interface.

[0121] The method for solving an optimization problem using a flexible modular approach may be implemented in various ways. One of many possible implementation embodiments called FOCUS, a flexible modular computational framework for solving constrained optimization problems is described herein.

[0122] FIG. 2 is a flowchart of a method for solving an optimization problem using the flexible modular computational framework, FOCUS, in accordance with some embodiments disclosed herein. More precisely, FIG. 2 shows the high-level procedure for solving an optimization problem using the flexible modular computational framework, FOCUS.

[0123] According to the processing operation 200, an indication of the optimization problem is obtained as an input. In some cases, the optimization problem may be any optimization problem described herein or any combination thereof. In some cases, the optimization problem is a combinatorial optimization problem. In some cases, the optimization problem is a constrained or unconstrained binary optimization problem. In some cases, the optimization problem is a constrained or unconstrained integer optimization problem. In some cases, the optimization problem is a constrained or unconstrained mixed-integer optimization problem. In some cases, the optimization problem is a constrained or unconstrained continuous optimization problem. In some cases, the optimization problem is a multi-objective optimization problem or an optimization problem under uncertainty.

[0124] FIG.3 is a flowchart of a procedure for providing an indication of the optimization problem, in accordance with some embodiments disclosed herein.

[0125] According to the processing operation 300, data representative of the solution space is provided. In some cases, the data representative is provided by a user. In some cases, data representative is extracted by the computer-implemented method.

[0126] For example, if the optimization problem is a travelling salesperson problem with n cities, the solution space comprises possible permutations of the n cities. The solution space size is then $n!$.

[0127] In another example wherein the vehicle routing optimization problem is provided with n cities and m vehicles, the solution space comprises possible assignments of the n cities to the m vehicles and possible permutations of the assigned cities to a given vehicle. The number of

configurations in the solution space equals $\binom{n}{n_1, n_2, \ldots n_m} n_1!\, n_2! \ldots n_m!$, where $n_1 + n_2 + \ldots + n_m = n, 0 \leq n_1, n_2, \ldots, n_m \leq n$.

[0128] Still referring to FIG. 3 and according to processing space 302, data representative of at least one objective function is provided.

[0129] In the travelling salesperson and the vehicle routing embodiments, the route length, and the sum of the lengths of the routes represent, in respective terms, the objective function.

[0130] According to processing operation 304, data representative of problem constraints is provided. In some cases, it may not be necessary to identify some or all of the problem constraints wherein the solution space is defined such that it may include only configurations that satisfy these constraints.

[0131] In classical routing examples such as travelling salesperson and vehicle routing problems, the problem constraints state that each city may be assigned to exactly one vehicle and each city may be visited exactly once. Defining the solution spaces as described in processing operation 300 satisfies the problem constraints by construction.

[0132] In the embodiment of the vehicle routing problem wherein there is an additional constraint that a set of cities cannot be served by the same vehicle, the solution space defined in processing operation 300 includes infeasible configurations. In this case, the problem constraints indicate the list of cities that cannot be served together.

[0133] Now referring back to FIG. 2 and according to the processing operation 202, it is determined if a custom interface is to be implemented for the optimization problem. In some cases, if the user is aiming for better performance, he/she may decide to incorporate the generation and the evaluation methods into the interface, otherwise, the interface may be a wrapper around generation and evaluation methods. If a custom interface is to be implemented, the procedure may continue to processing operation 208, if not, it may continue to processing operation 204.

[0134] According to processing operation 204, an indication of the generation method for generating at least one configuration and at least one allowed move is provided as an input. In some cases, the generation method may be provided by a user. In some cases, the generation method is provided by the computer-implemented method. In some cases, the generation method may comprise procedures that describe how to generate a configuration and a move, and how to transform a given configuration into a new configuration using a move.

[0135] FIG.4 is a flowchart of a procedure for initializing a generation method, in accordance with some embodiments disclosed herein.

[0136] After identifying the solution space for the optimization problem, according to the processing operation 400, a decision may be made whether one of the pre-implemented generation methods in the flexible modular computational framework FOCUS can be used.

[0137] If one of the pre-implemented generation methods can be used, the generation method for the given optimization problem may be initialized with the selected generation method. In the embodiment wherein the optimization problem is a travelling salesperson problem example, the solution space is represented by a permutation and the PermutationGenerator in the flexible modular computational framework FOCUS can be used with no adjustment.

[0138] If none of the pre-implemented generation methods can be used, the procedure continues to processing operation 402. In the embodiment wherein the optimization problem is a vehicle routing problem, a custom generation method may be provided.

[0139] Still referring to FIG. 4 and according to processing operation 402, the custom generation method is provided by a procedure to generate a configuration x from the solution space D.

[0140] For the vehicle routing problem example, the following procedure generates a random configuration.

1. Initialize a list x containing an empty list for each vehicle
2. Randomize the list of n cities
3. Mark the cities as unvisited
4. While there is at least one city unvisited
    a. Pick a vehicle randomly
    b. Assign the city to the selected vehicle
    c. Make the city visited
5. Randomly shuffle the cities assigned to each vehicle
6. Return x as the random configuration

[0141] The configuration $x = [[4, 7, 1, 5, 8], [2, 6, 3]]$ represents an example generated by the above procedure indicating that the cities $4, 7, 1, 5$, and 8 are served by the first vehicle and the second vehicle serves cities $2, 6$, and 3.

[0142] Still referring to FIG. 4 and according to processing operation 404, the custom generation method is provided by procedures to generate at least one move in order for the optimization algorithm to navigate through the search space.

[0143] For the vehicle routing problem, a move may be drawn as below:

1. Given the current configuration x, pick a random city i from 1 to n
2. Pick a random vehicle j that currently does not serve city i
3. Pick a random position k in vehicle j's list of cities

[0144] Iterating through the above procedure for a given number of times results in a number of moves.

[0145] In some cases, there are various ways to define a move for a given optimization problem.

[0146] In some cases, it may be sufficient to provide only one procedure to define a custom generation method in the flexible modular computational framework FOCUS. For the vehicle routing problem, examples of different moves may comprise:

- swapping two cities assigned to two different vehicles
- swapping the positions of two cities assigned to the same vehicle

[0147] Still referring to FIG. 4 and according to processing operation 406, given the procedure for generating a move, a procedure of applying a given move to a given configuration is provided in order to navigate to a new configuration.

[0148] For the vehicle routing problem, representing the move as (i, j, k) and the current configuration x, the procedure in processing operation 406 may be:

Remove city i from its current vehicle and assign it to the k-th position in vehicle j's list of cities.

[0149] Now referring back to FIG. 2 and according to processing operation 206, an indication of the evaluation method for evaluating the objective functions and the constraints for a given configuration is provided as an input. In some cases, the evaluation method may be provided by a user. In some cases, the evaluation method is provided by the computer-implemented method. In some cases, the evaluation method may comprise procedures that evaluate at least one objective function and at least zero constraint for a given configuration and a move.

[0150] FIG.5 is a flowchart of a procedure for initializing an evaluation method, in accordance with some embodiments disclosed herein.

[0151] According to processing operation 500, the evaluation method is provided by a procedure to calculate objective functions' value and to evaluate constraints for a given configuration x. The constraints for the configuration x are evaluated to determine the violation of each constraint in $C(x)$ and the feasibility of the configuration x.

[0152] In the travelling salesperson problem embodiment, the objective function value for the configuration x is equal to the sum of the distances travelled from city $x[i]$ to city $x[i+1]$ where $i = 0, 1, 2, \ldots, n, x[0] = x[n+1] = depot$. In this embodiment and since the solution space may include feasible configurations, there is no constraint to be evaluated for a configuration and constraints evaluation results are irrelevant.

[0153] In the vehicle routing problem embodiment with the additional constraint that cities i and j cannot be served by the same vehicle, the objective function value for a given configuration x

equals to the sum of the route lengths over vehicles wherein the route of each vehicle is calculated following the same procedure as in the travelling salesperson problem.

[0154] In the configurations where i and j are assigned to different vehicles, the constraint evaluates to feasibility and zero violation. In the configurations where i and j are assigned to the same vehicle, the constraint evaluates to infeasibility and has a violation of one.

[0155] According to processing operation 502, a procedure is provided to the evaluation method to accept the new configuration x and update the internal state of the evaluation method.

[0156] More precisely, given the current configuration x and its corresponding objective function value and constraints evaluation, the internal state of the evaluation method including the reference configuration, the reference objective function value, and the reference constraints evaluation are updated.

[0157] According to processing operation 504, a procedure to calculate the objective functions' value and quantify the constraints evaluation for a given move is provided. In some cases, the evaluation method always interprets a move relative to its current internal state.

[0158] In the travelling salesperson problem embodiment, a move may be prescribed by swapping cities i and j in the current configuration $x = [\dots k, i, l, \dots, u, j, v, \dots]$. The objective function value for the new configuration can be obtained by updating the reference objective function f. The new objective function value is equal to $f - (d[k][i] - d[i][l] - d[u][j] - d[j][v]) + (d[k][j] + d[j][l] + d[u][i] + d[i][v])$ where d is the distance matrix. In this example, there is no constraint to be evaluated for the move and the constraints evaluation results are irrelevant.

[0159] Still referring to FIG. 5 and according to processing operation 506, a procedure to accept a move and update the internal state of the evaluation method is provided to the evaluation method.

[0160] Considering the same embodiment as in processing operation 504, the procedure in processing operation 506 first swaps cities i and j in $x = [\dots k, i, l, \dots, u, j, v, \dots]$ resulting in $x_{new} = [\dots k, j, l, \dots, u, i, v, \dots]$. It then applies the same method as in the processing operation 502 to update the internal state of the evaluation method.

[0161] Now referring back to FIG. 2 and according to processing operation 208, an indication of the interface is provided as an input. In some cases, the interface acts as a layer between algorithms and generation and evaluation methods.

[0162] In some cases, the generation and the evaluation methods are wrapped into an interface interacting with at least one algorithm to draw and evaluate configurations and moves used by an algorithm. More precisely, the interface comprises methods to generate configurations and moves, to evaluate a set of objective functions and constraints for a given configuration and

move. Wrapping the generation and evaluation methods by the interface is modular and flexible but it may have some overhead. In some cases, for better performance, the user may decide to directly incorporate the methods for generation and evaluation of configurations and moves into the interface. In some cases, the methods included in the interface may deal with the constraints when the interface is acting as an abstraction layer between algorithms and optimization problems.

[0163] In some cases, constraints handling procedures may be embedded in the interface to guide the search toward feasible configurations. In some cases, the search space includes both feasible and infeasible configurations, and the generation method does not necessarily draw feasible configurations or moves from the search space. Constraints handling procedures may be of various types. The list of constraints handling procedures includes but is not limited to incorporating static and adaptive penalty terms into the objective functions, rejection sampling, rejection of infeasible configurations, and extending the search space to include Lagrange multipliers, or embedding techniques such as constraint programming, linear programming, and mixed-integer programming in the search space to restrict the search space.

[0164] FIG.6 is a flowchart of a procedure for integrating a generation method, an evaluation method, and a constraints handling procedure into an interface, in accordance with some embodiments disclosed herein.

[0165] According to processing operation 600, a procedure for handling the constraints is provided considering the provided generation and the evaluation methods.

[0166] In some cases, the search space includes only feasible solutions, and a procedure for handling constraints may not be needed as all configurations are feasible. In some cases, the search space comprises infeasible solutions, and an interface that deals with constraints may be chosen. In some cases, the interface may be of various types. In some cases, constraints handling procedures may be chosen to be embedded into the interface. The constraints handling procedures may be any constraints handling procedures, such as any constraints handling procedures described herein. In some cases, constraints handling procedures comprise rejection sampling, static penalty terms, and adaptive penalty terms.

[0167] Still referring to FIG. 6 and according to processing operation 602, it may be determined whether a pre-implemented interface in the flexible modular computational framework FOCUS can be used or not.

[0168] In some cases, if a generation method and an evaluation method are not provided, a custom interface may be implemented, and the procedure continues to processing operation 604.

[0169] In case a custom interface is added, according to the processing operation 604, it may be provided by indications of:

- procedures in the generation method for generating configurations and moves
- procedures in the evaluation method for quantifying the objective functions' values and evaluating the constraints for a given configuration and move

[0170] Now referring back to FIG. 2 and according to processing operation 210, at least one optimization algorithm among the suite of the pre-implemented algorithms is selected. In some cases, the algorithm is selected by a user. In some cases, the selected algorithms may be of various types, it may be any algorithm, such as any algorithm described herein. In some cases, the suite of the pre-implemented algorithms is a group of algorithms consisting of local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.

[0171] In some cases, there is no limit on the number of the selected algorithms. The selected algorithms utilize the generation method and the evaluation method to generate the configurations and the allowed moves and to evaluate the set of objective functions and constraints for a given configuration. The user may select available pre-implemented optimization algorithms in the flexible modular computational framework FOCUS to solve the provided optimization problem with no additional modification to the evaluation, generation, and interface methods.

[0172] Still referring to FIG. 2 and according to processing operation 212, after the optimization algorithms are selected, the parameters of each algorithm are set. In some cases, the number of parameters and the procedures for setting their values may be different for various optimization algorithms.

[0173] In some cases, wherein the selected optimization algorithm is simulated annealing, the user may set the parameters such as number of independent calls to the algorithm, number of Monte Carlo operations for each run, and the annealing schedule.

[0174] In some cases, wherein the selected optimization algorithm is tabu search, the user may set the number of independent calls to the algorithm, the number of iterations, and the length of tabu tenure.

[0175] Still referring to FIG. 2 and according to processing operation 214, the generation method, the evaluation method, interface, and the selected algorithms are provided to at least one computational platform comprising at least one processor. In some cases, the computational platform may be of various types. The computational platform may be of any suitable computational platform such as any computational platform described herein with respect to the system shown in FIG. 7. In some cases, the computational platform comprises a field-

programmable gate array (FPGA), In some cases, the computational platform comprises an application-specific integrated circuit (ASIC). In some cases, the computational platform comprises central processing unit (CPU). In some cases, the computational platform comprises a graphics processing unit (GPU). In some cases, the computational platform comprises a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a digital annealer, a quantum annealer or an optical quantum computer. In some cases, the user is required to specify a computational platform to be used to run each of the procedures in the generation, the evaluation, and/or the interface methods and the selected optimization algorithms.

[0176] According to the processing operation 216, each algorithm is executed independently on the corresponding computational platform until a stopping criterion or a set of stopping criteria is met.

[0177] In some cases, wherein the selected algorithm is simulated annealing, each independent call to the algorithm stops after performing a given number of Monte Carlo operations. In some cases, wherein the selected algorithm is tabu search, the run terminates after reaching a given number of iterations.

[0178] Still referring to FIG. 2 and according to processing operation 218, after each algorithm terminates its runs, at least one solution comprising a configuration and its corresponding objective functions' values and constraints evaluation is provided.

[0179] According to the processing operation 220, the result is obtained. The result represents the best feasible solution among the solutions collected in processing operation 218.

[0180] In case, there is no feasible solution among the solutions obtained in the processing operation 218, null is provided as the result.

[0181] FIG. 7 is a diagram of a system for solving an optimization problem using a flexible modular approach, in accordance with some embodiments disclosed herein. The system comprises a digital computer 700 comprising at least one processing device 704 and a memory 710 comprising a computer program executable by the processing device to obtain an indication of an optimization problem; to providing data representative of a search space comprising configurations, of at least one objective function and of at least zero constraints using the optimization problem; to providing an indication of at least one generation method for generating at least one configuration and at least one allowed move; to providing an indication of at least one evaluation method for evaluating the at least one objective function and the at least zero constraints for a given configuration; to executing each of selected algorithms; to providing results comprising at least one solution to the optimization problem. The memory 710 further comprising implementation of at least zero optimization problems, objective functions, search spaces, constraints, moves, optimization algorithms, interfaces, generation methods and

evaluation methods. In some cases, the digital computer 700 may be of various types, such as any digital computer disclosed herein.

[0182] The system further comprises at least one computational platform 702. The computational platform 702 comprises at least one processing unit. In some cases, the at least one processing unit 714 may be of various types such as any processing unit disclosed herein. More precisely, the at least one processing unit may comprise at least one member of the group of hardware consisting of FPGA, ASIC, quantum annealer, digital annealer, GPU, TSP and TPU. In some cases, each of the hardware may be used as part of the system, to execute the whole method, or any part of it, alone, or in combination with other hardware. In some cases, the hardware may be used for: evaluation of the objective functions given an initial configuration; evaluation of the constraints given an initial configuration; evaluation of the objective functions given a configuration and its objective functions' values, and a prospective move; evaluation of the constraints given a configuration and its objective functions' values, and a prospective move; execution of an optimization algorithm or a part of an optimization algorithm; generation of random numbers; calculation of Boltzmann factors for prospective moves; generation of new moves; generation of random solutions; execution of functions of the interface, including a part or all of the above.

[0183] The computational platform 702 is operatively connected to the digital computer 700. The computational platform further comprises the read-out control system 716.

[0184] While preferred embodiments of the present invention have been shown and described herein, it will be obvious to those skilled in the art that such embodiments are provided by way of example only. Numerous variations, changes, and substitutions will now occur to those skilled in the art without departing from the invention. It should be understood that various alternatives to the embodiments of the invention described herein may be employed in practicing the invention. It is intended that the following claims define the scope of the invention and that methods and structures within the scope of these claims and their equivalents be covered thereby.

CLAIMS:

WHAT IS CLAIMED IS:

1. A method for solving an optimization problem inputted by a user, comprising:

(a) using an interface of a digital computer to receive an indication of:

(i) said optimization problem,

(ii) one or more optimization problem specifications of a plurality of optimization problem specifications,

(iii) at least one generation method and at least one evaluation method,

(iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and

(v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms;

(b) transmitting a request to said one or more computational platforms to execute said one or more algorithms, wherein upon execution of said one or more algorithms by said one or more computational platforms, said one or more computational platforms yields a solution to said optimization problem; and

(c) providing said solution to said optimization problem from said computational platform at said interface.

2. The method of claim 1, wherein said interface is a user interface (UI).
3. The method of claim 2, wherein said UI is a graphical UI (GUI).
4. The method of claim 1, wherein said one or more optimization problem specifications comprises at least one of: a search space comprising configurations of a plurality of search spaces, one or more objective functions of a plurality of objective functions, an allowed move of a plurality of allowed moves, or at least zero constraints of a plurality of constraints.
5. The method of claim 1 or 4, wherein (a)(ii) comprises on said interface (1) presenting at least a subset of said plurality of optimization problem specifications to said user and (2) receiving a selection of said one or more optimization problem specifications from said user.
6. The method of claim 5, further comprising receiving a selection of said one or more optimization problem specifications from said user, wherein said selection comprises one

or more of a selected search space comprising configurations, a selected allowed move, one or more selected objective functions, and at least one selected constraint.

7. The method of claim 1, wherein (a)(iii) comprises on said interface (1) presenting a plurality of generation methods and a plurality of evaluation methods to said user and (2) receiving a selection of said at least one generation method and said at least one evaluation method from said user.
8. The method of claim 7, wherein said at least one generation method generates at least one configuration from a search space and wherein said at least one generation method generates at least one allowed move within said search space.
9. The method of claim 7, wherein said at least one evaluation method evaluates one or more objective functions and determines that at least zero constraints are satisfied for a given configuration.
10. The method of claim 1, wherein (a)(iv) comprises on said interface (1) presenting at least a subset of said plurality of algorithms to said user and (2) receiving a selection of said one or more algorithms from said user.
11. The method of claim 1, wherein (a)(iv) comprises on said interface (1) presenting at least a subset of said plurality of computational platforms to said user and (2) receiving a selection of said one or more computational platforms from said user.
12. The method of claim 5, 7, 10, or 11, wherein (a)(iv) comprises aiding or suggesting said selection by said interface.
13. The method of claim 1, wherein (b) comprises executing said one or more algorithms until a stopping criterion is met on said one or more computational platforms.
14. The method of claim 1, wherein said interface is configured to communicate with said plurality of computational platforms over a network.
15. The method of claim 1, wherein said one or more optimization specifications further comprises at least one constraints handling procedure.
16. The method of claim 15, wherein said at least one constraints handling procedure comprises at least one of: incorporating static and adaptive penalty terms into said one or more objective functions, rejection sampling or rejection of infeasible configurations, extending the search space to include Lagrange multipliers, or embedding techniques, optionally, wherein said embedding techniques comprise one or more of constraint programming, linear programming, and mixed-integer programming in said search space to restrict said search space.
17. The method of claim 14 or 15, wherein said one or more algorithms employs said at least one constraints handling procedure.

18. The method of claim 1, wherein said one or more algorithms comprises one or more of: local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.
19. The method of claim 4, wherein said search space comprises one or more of: a search space comprising binary variables, a search space comprising integer variables, a search space comprising categorical variables, a search space comprising permutations, a search space comprising continuous variables.
20. The method of claim 1, wherein said one or more computational platforms comprises one or more of: a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), central processing unit (CPU), graphics processing unit (GPU), a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a quantum annealer, integrated photonic coherent Ising machine, or an optical quantum computer.
21. The method of claim 1, wherein said optimization problem comprises one or more of: combinatorial optimization, constrained and unconstrained binary optimization, constrained and unconstrained integer optimization, constrained and unconstrained mixed-integer optimization, constrained and unconstrained continuous optimization, multi-objective optimization, or optimization under uncertainty.
22. The method of claim 4, wherein said one or more objective functions comprise one or more of: a linear objective function, a quadratic objective function, a non-linear objective function, a non-linear convex objective function, a non-linear non-convex objective function, a look-up table objective function, or a black-box objective function.
23. The method of claim 4, wherein said at least zero constraints comprises one or more of: a linear equality constraint, a linear inequality constraint, a quadratic constraint, a non-linear parametric constraint, a look-up table constraint, or a black-box constraint.
24. A system for solving an optimization problem comprising:

an interface of a digital computer, said digital computer operatively coupled to a memory comprising instructions, wherein said digital computer is configured to execute said instructions to at least: receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said

optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms,

wherein said interface is in communication with a computational platform configured to execute said one or more algorithms, to yield a solution to said optimization problem, wherein said solution to said optimization problem from said computational platform is provided at said interface.

25. The system of claim 24, wherein said interface is a user interface (UI).
26. The system of claim 25, wherein said UI is a graphical UI (GUI).
27. The system of claim 24, wherein said one or more optimization problem specifications comprises at least one of: a search space comprising configurations of a plurality of search spaces, one or more objective functions of a plurality of objective functions, an allowed move of a plurality of allowed moves, or at least zero constraints of a plurality of constraints.
28. The system of claim 24 or 27, wherein said interface is configured to (1) present at least a subset of said plurality of optimization problem specifications to said user and (2) receive a selection of said one or more optimization problem specifications from said user.
29. The system of claim 28, wherein said interface is configured to receive a selection of said one or more optimization problem specifications from said user, wherein said selection comprises one or more of a selected search space comprising configurations, a selected allowed move, one or more selected objective functions, and at least one selected constraint.
30. The system of claim 24, wherein said interface is configured to (1) present a plurality of generation methods and a plurality of evaluation methods to said user and (2) receive a selection of said at least one generation method and said at least one evaluation method from said user.
31. The system of claim 24, wherein said at least one generation method is configured to generate at least one configuration from a search space and wherein said at least one generation method generates at least one allowed move within said search space.
32. The system of claim 24, wherein said at least one evaluation method is configured to evaluate one or more objective functions and determines that at least zero constraints are satisfied for a given configuration.

33. The system of claim 24, wherein said interface is configured to (1) present at least a subset of said plurality of algorithms to said user and (2) receive a selection of said one or more algorithms from said user.
34. The system of claim 24, wherein said interface is configured to (1) present at least a subset of said plurality of computational platforms to said user and (2) receive a selection of said one or more computational platforms from said user.
35. The system of claim 28, 30, 33, or 34, wherein said selection is aided or suggested by said interface.
36. The system of claim 24, wherein said digital computer is configured to execute said instructions to execute said one or more algorithms until a stopping criterion is met on said one or more computational platforms.
37. The system of claim 24, wherein said interface is configured to communicate with said plurality of computational platforms over a network.
38. The system of claim 24, wherein said one or more optimization specifications further comprises at least one constraints handling procedure.
39. The system of claim 38, wherein said at least one constraints handling procedure comprises at least one of: incorporating static and adaptive penalty terms into said one or more objective functions, rejection sampling or rejection of infeasible configurations, or extending the search space to include Lagrange multipliers, or embedding techniques such as constraint programming, linear programming, and mixed-integer programming in said search space to restrict said search space.
40. The system of claim 38 or 39, wherein said one or more algorithms employs said at least one constraints handling procedure.
41. The system of claim 24, wherein said one or more algorithms comprises one or more of: local search heuristic algorithms, simulated annealing, simulated quantum annealing, parallel tempering, population annealing, constrained population annealing, exhaustive search, greedy search, tabu search, path relinking, constrained simulated annealing, constrained parallel tempering, evolutionary search, or machine learning-based algorithms.
42. The system of claim 27, wherein said search space comprises one or more of: a search space comprising binary variables, a search space comprising integer variables, a search space comprising categorical variables, a search space comprising permutations, a search space comprising continuous variables.
43. The system of claim 24, wherein said one or more computational platforms comprises one or more of: a field-programmable gate array (FPGA), an application-specific integrated

circuit (ASIC), central processing unit (CPU), graphics processing unit (GPU), a tensor processing unit (TPU), a tensor streaming processor (TSP), a quantum computer, a quantum annealer, integrated photonic coherent Ising machine, or an optical quantum computer.

44. The system of claim 24, wherein said optimization problem comprises one or more of: combinatorial optimization, constrained and unconstrained binary optimization, constrained and unconstrained integer optimization, constrained and unconstrained mixed-integer optimization, constrained and unconstrained continuous optimization, multi-objective optimization, or optimization under uncertainty.
45. The system of claim 27, wherein said one or more objective functions comprise one or more of: a linear objective function, a quadratic objective function, a non-linear objective function, a non-linear convex objective function, a non-linear non-convex objective function, a look-up table objective function, or a black-box objective function.
46. The system of claim 27, wherein said at least zero constraints comprises one or more of: a linear equality constraint, a linear inequality constraint, a quadratic constraint, a non-linear parametric constraint, a look-up table constraint, or a black-box constraint.
47. A non-transitory computer readable medium comprising machine-executable code that upon execution by a digital computer in communication with a computational platform implements a method for solving an optimization problem from a user, comprising:
    a. using an interface of a digital computer to receive an indication of (i) said optimization problem, (ii) one or more optimization problem specifications of a plurality of optimization problem specifications, (iii) at least one generation method and at least one evaluation method, (iv) one or more algorithms of a plurality of algorithms, wherein said one or more algorithms is configured to solve said optimization problem subject to at least said one or more optimization problem specifications using said at least one generation method and said at least one evaluation method, and (v) one or more computational platforms of a plurality of computational platforms, wherein said one or more computational platforms is configured to execute said one or more algorithms;
    b. using said interface to transmit a request to said one or more computational platforms to execute said one or more algorithms to yield a solution to said optimization problem; and
    c. providing said solution to said optimization problem from said one or more computational platforms at said interface.

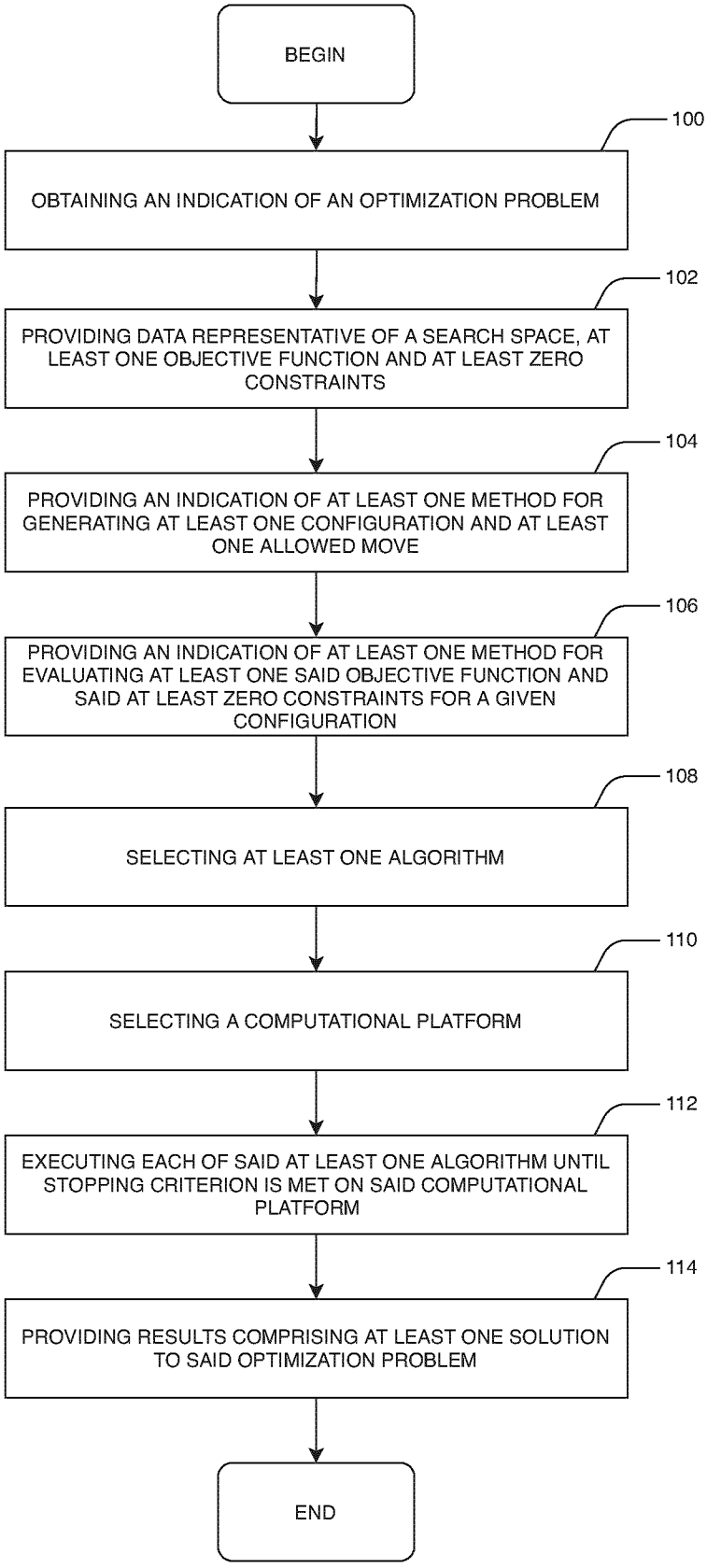
BEGIN
100
OBTAINING AN INDICATION OF AN OPTIMIZATION PROBLEM
102
PROVIDING DATA REPRESENTATIVE OF A SEARCH SPACE, AT LEAST ONE OBJECTIVE FUNCTION AND AT LEAST ZERO CONSTRAINTS
104
PROVIDING AN INDICATION OF AT LEAST ONE METHOD FOR GENERATING AT LEAST ONE CONFIGURATION AND AT LEAST ONE ALLOWED MOVE
106
PROVIDING AN INDICATION OF AT LEAST ONE METHOD FOR EVALUATING AT LEAST ONE SAID OBJECTIVE FUNCTION AND SAID AT LEAST ZERO CONSTRAINTS FOR A GIVEN CONFIGURATION
108
SELECTING AT LEAST ONE ALGORITHM
110
SELECTING A COMPUTATIONAL PLATFORM
112
EXECUTING EACH OF SAID AT LEAST ONE ALGORITHM UNTIL STOPPING CRITERION IS MET ON SAID COMPUTATIONAL PLATFORM
114
PROVIDING RESULTS COMPRISING AT LEAST ONE SOLUTION TO SAID OPTIMIZATION PROBLEM
END

Figure 1

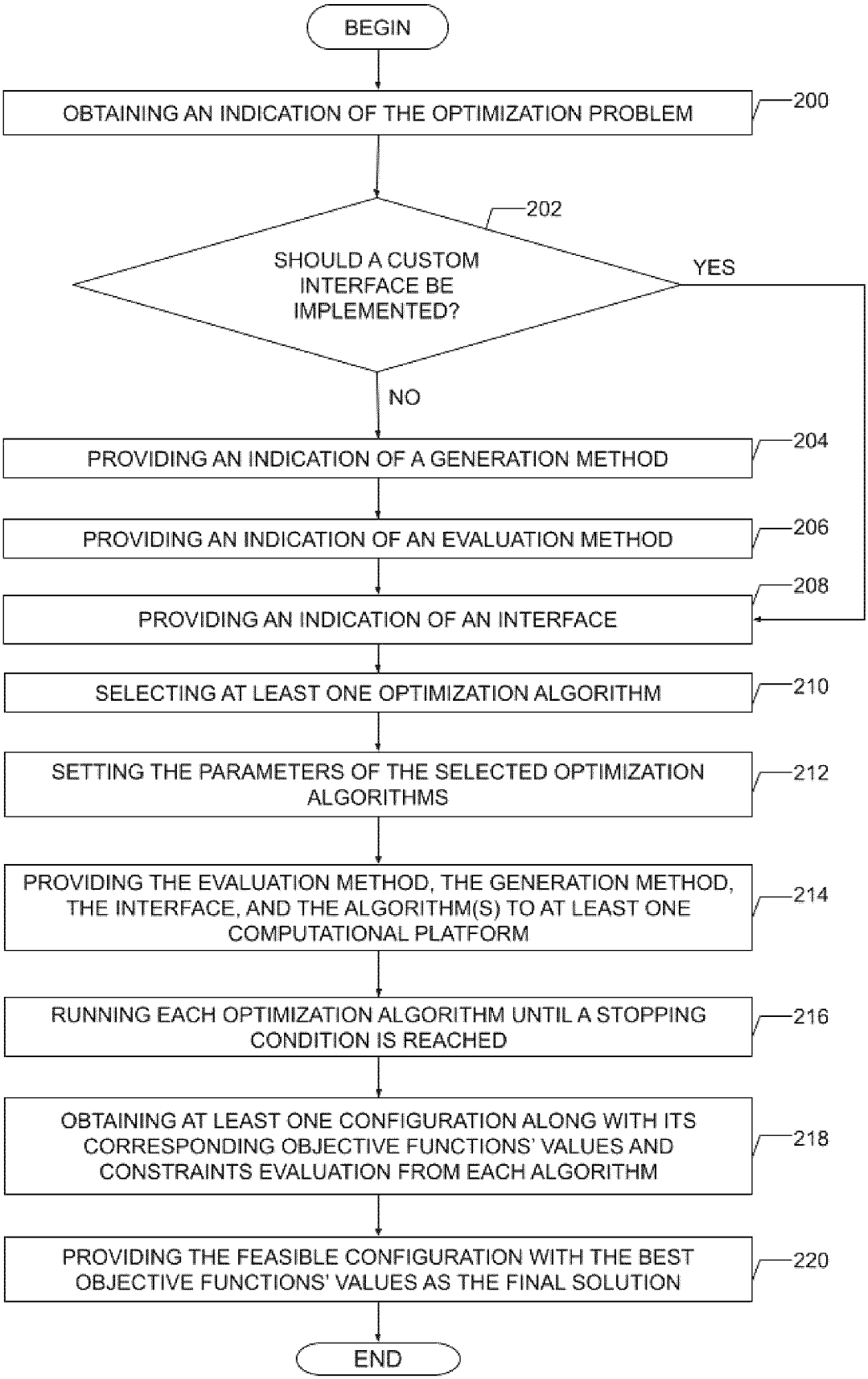
BEGIN
OBTAINING AN INDICATION OF THE OPTIMIZATION PROBLEM
200
202
SHOULD A CUSTOM INTERFACE BE IMPLEMENTED?
YES
NO
PROVIDING AN INDICATION OF A GENERATION METHOD
204
PROVIDING AN INDICATION OF AN EVALUATION METHOD
206
PROVIDING AN INDICATION OF AN INTERFACE
208
SELECTING AT LEAST ONE OPTIMIZATION ALGORITHM
210
SETTING THE PARAMETERS OF THE SELECTED OPTIMIZATION ALGORITHMS
212
PROVIDING THE EVALUATION METHOD, THE GENERATION METHOD, THE INTERFACE, AND THE ALGORITHM(S) TO AT LEAST ONE COMPUTATIONAL PLATFORM
214
RUNNING EACH OPTIMIZATION ALGORITHM UNTIL A STOPPING CONDITION IS REACHED
216
OBTAINING AT LEAST ONE CONFIGURATION ALONG WITH ITS CORRESPONDING OBJECTIVE FUNCTIONS' VALUES AND CONSTRAINTS EVALUATION FROM EACH ALGORITHM
218
PROVIDING THE FEASIBLE CONFIGURATION WITH THE BEST OBJECTIVE FUNCTIONS' VALUES AS THE FINAL SOLUTION
220
END

Figure 2

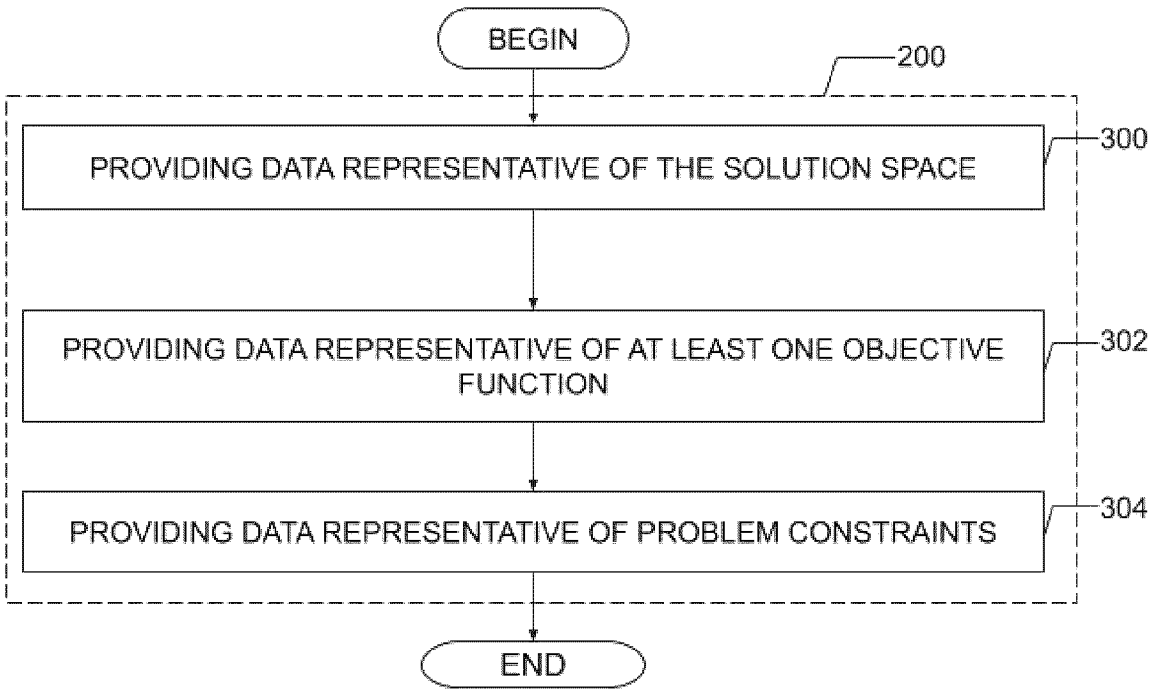
BEGIN
200
PROVIDING DATA REPRESENTATIVE OF THE SOLUTION SPACE
300
PROVIDING DATA REPRESENTATIVE OF AT LEAST ONE OBJECTIVE FUNCTION
302
PROVIDING DATA REPRESENTATIVE OF PROBLEM CONSTRAINTS
304
END

Figure 3

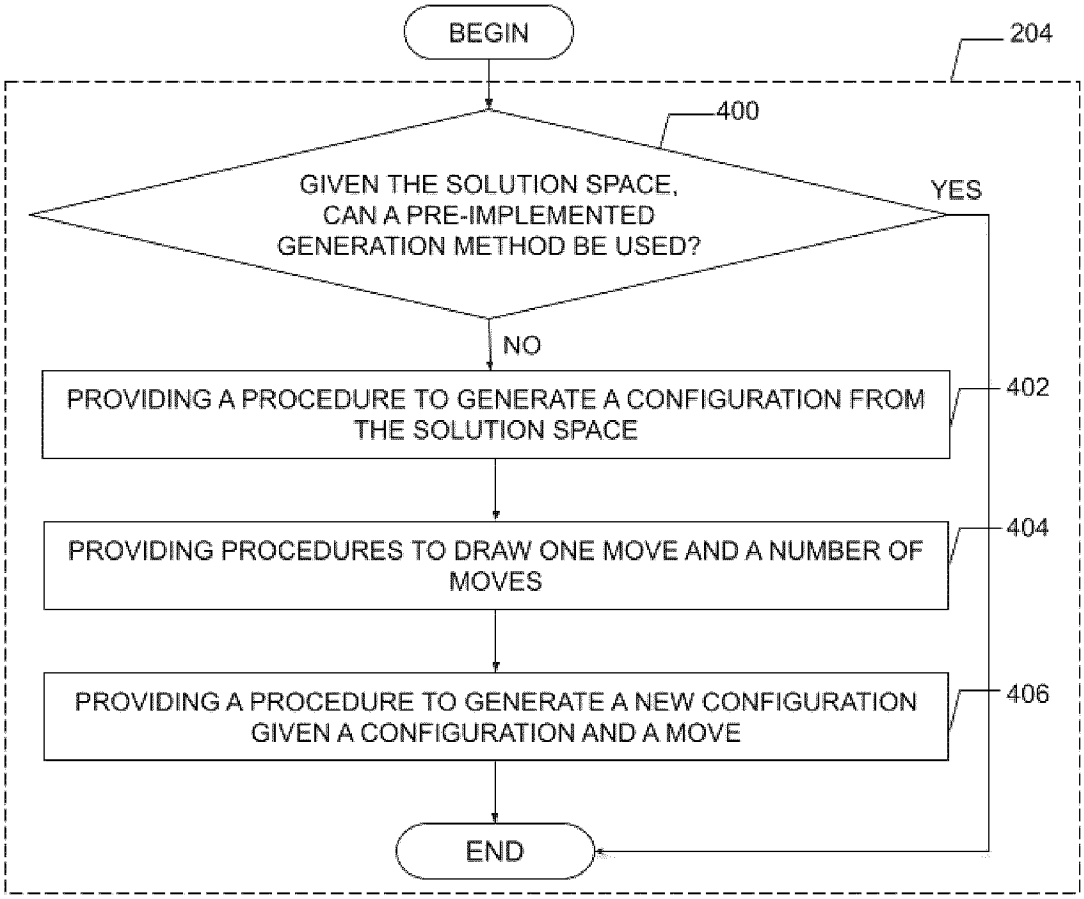
BEGIN
204
400
GIVEN THE SOLUTION SPACE, CAN A PRE-IMPLEMENTED GENERATION METHOD BE USED?
YES
NO
PROVIDING A PROCEDURE TO GENERATE A CONFIGURATION FROM THE SOLUTION SPACE
402
PROVIDING PROCEDURES TO DRAW ONE MOVE AND A NUMBER OF MOVES
404
PROVIDING A PROCEDURE TO GENERATE A NEW CONFIGURATION GIVEN A CONFIGURATION AND A MOVE
406
END

Figure 4

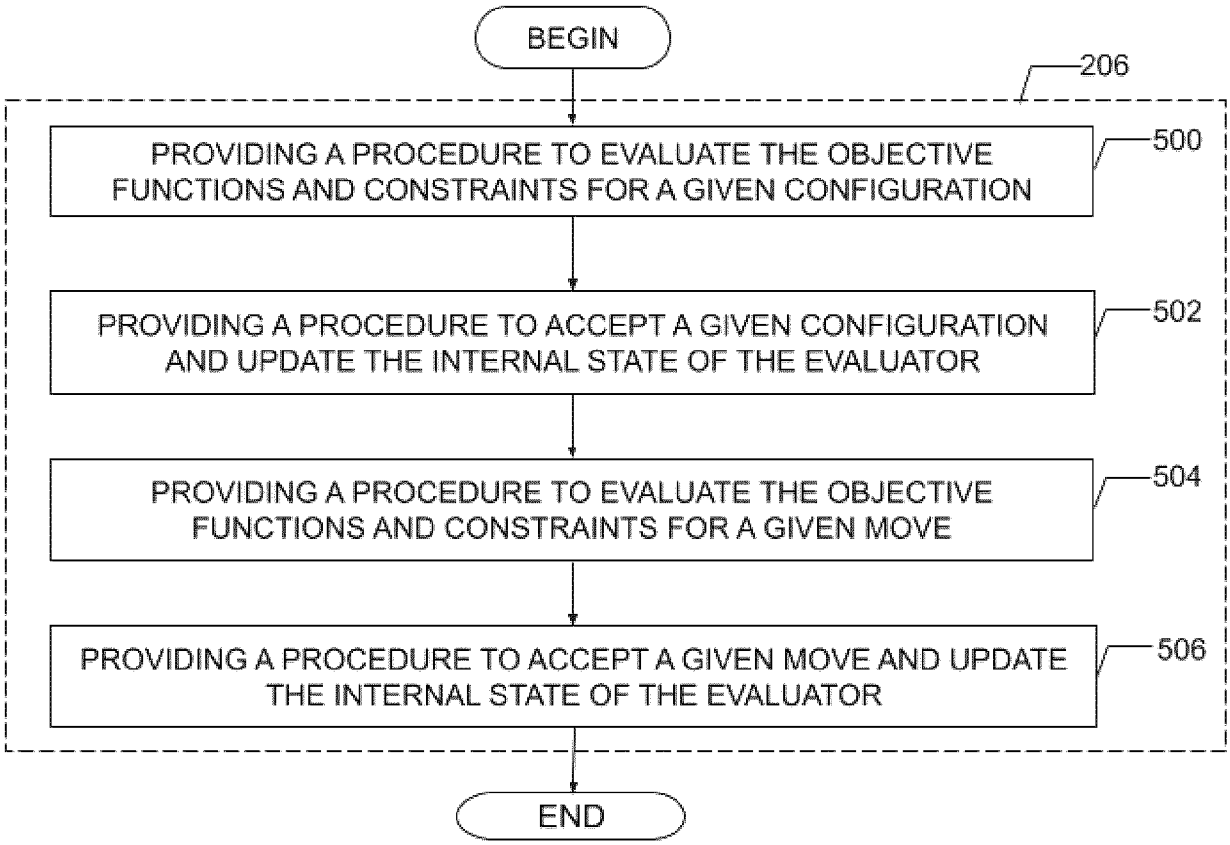
BEGIN
206
PROVIDING A PROCEDURE TO EVALUATE THE OBJECTIVE FUNCTIONS AND CONSTRAINTS FOR A GIVEN CONFIGURATION
500
PROVIDING A PROCEDURE TO ACCEPT A GIVEN CONFIGURATION AND UPDATE THE INTERNAL STATE OF THE EVALUATOR
502
PROVIDING A PROCEDURE TO EVALUATE THE OBJECTIVE FUNCTIONS AND CONSTRAINTS FOR A GIVEN MOVE
504
PROVIDING A PROCEDURE TO ACCEPT A GIVEN MOVE AND UPDATE THE INTERNAL STATE OF THE EVALUATOR
506
END

Figure 5

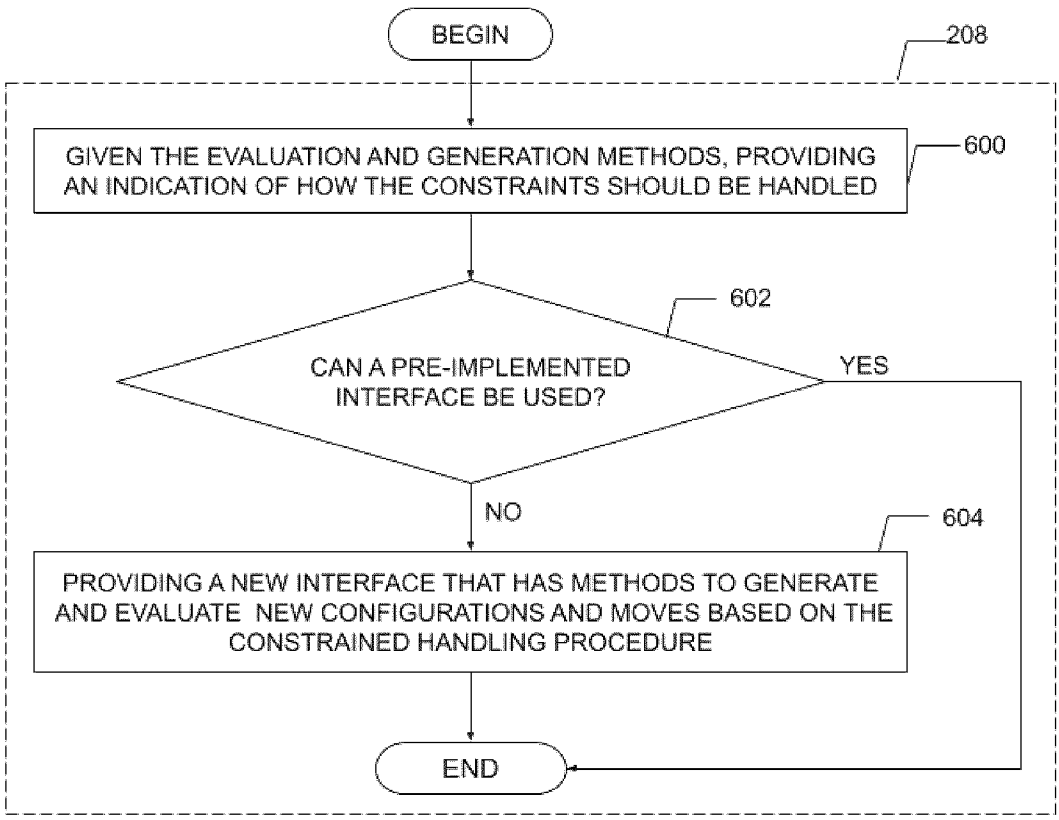
BEGIN
208
GIVEN THE EVALUATION AND GENERATION METHODS, PROVIDING AN INDICATION OF HOW THE CONSTRAINTS SHOULD BE HANDLED
600
602
CAN A PRE-IMPLEMENTED INTERFACE BE USED?
YES
NO
604
PROVIDING A NEW INTERFACE THAT HAS METHODS TO GENERATE AND EVALUATE NEW CONFIGURATIONS AND MOVES BASED ON THE CONSTRAINED HANDLING PROCEDURE
END

Figure 6

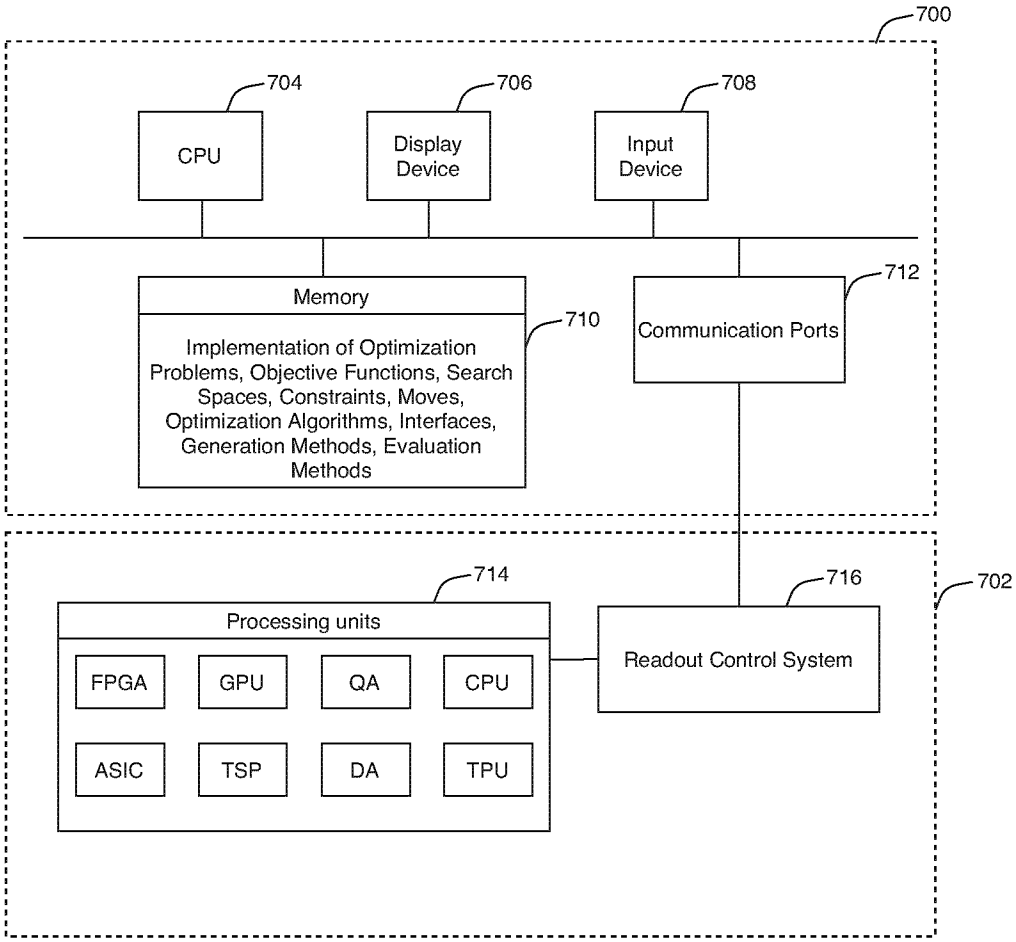
700
704
CPU
706
Display Device
708
Input Device
Memory
710
Implementation of Optimization Problems, Objective Functions, Search Spaces, Constraints, Moves, Optimization Algorithms, Interfaces, Generation Methods, Evaluation Methods
712
Communication Ports
714
Processing units
FPGA
GPU
QA
CPU
ASIC
TSP
DA
TPU
716
Readout Control System
702

Figure 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2021/050709

A. CLASSIFICATION OF SUBJECT MATTER
IPC: **G06F 17/10** (2006.01) , ***G06F 17/11*** (2006.01) , ***G06F 3/048*** (2013.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: G06F 17/10 (2006.01) , G06F 17/11 (2006.01) , G06F 3/048 (2013.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Databases: Questel Orbit; Google Patents
Keywords: optimization; algorithm+; search; space; objective; function; constraint+; objective; function; solving; problem; computation; configuration

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US20200143910 A1, Noori et al., 07 May 2020 (07-05-2020) * paragraphs [0029-0255], Fig. 3	1-47
A	US20150006443 A1, Rose et al., 01 January 2015 (01-01-2015) * Abstract; claims 1-3; paragraphs [0002-0315]	1-47
A	US20050187844 A1, Chalermkraivuth et al., 25 August 2005 (25-08-2005) * Abstract; claim 1; paragraphs [0021-0023, 0094-0118]	1-47
A	McGeoch et al., "Experimental evaluation of an adiabiatic quantum system for combinatorial optimization", CF '13: Proceedings of the ACM International Conference on Computing Frontiers, 31 May 2013 (31-05-2013), Article No.: 23, https://doi.org/10.1145/2482767.2482797 * Pages 1–11	1-47

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"D"	document cited by the applicant in the international application	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"E"	earlier application or patent but published on or after the international filing date	"&"	document member of the same patent family
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 19 July 2021 (19-07-2021)	Date of mailing of the international search report 03 August 2021 (03-08-2021)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 819-953-2476	Authorized officer Leslie Yeow (819) 639-8372

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CA2021/050709

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2020143910A1	07 May 2020 (07-05-2020)	CA3060810A1	02 May 2020 (02-05-2020)
US2015006443A1	01 January 2015 (01-01-2015)	US9727824B2	08 August 2017 (08-08-2017)
		CN105531725A	27 April 2016 (27-04-2016)
		CN105531725B	13 March 2018 (13-03-2018)
		CN108256651A	06 July 2018 (06-07-2018)
		EP3014534A1	04 May 2016 (04-05-2016)
		EP3014534A4	22 March 2017 (22-03-2017)
		JP2016531343A	06 October 2016 (06-10-2016)
		JP6465876B2	06 February 2019 (06-02-2019)
		JP2019096334A	20 June 2019 (20-06-2019)
		JP6718994B2	08 July 2020 (08-07-2020)
		US2016321559A1	03 November 2016 (03-11-2016)
		US10318881B2	11 June 2019 (11-06-2019)
		US2017351974A1	07 December 2017 (07-12-2017)
		WO2014210368A1	31 December 2014 (31-12-2014)
US2005187844A1	25 August 2005 (25-08-2005)	US7542932B2	02 June 2009 (02-06-2009)
		WO2005081902A2	09 September 2005 (09-09-2005)
		WO2005081902A3	18 May 2007 (18-05-2007)