Received 30 May 2022, accepted 13 July 2022, date of publication 21 July 2022, date of current version 28 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3193143

RESEARCH ARTICLE

Exploiting Hardware and Software Advances for Quadratic Models of Wind Farm Layout Optimization

ARIK SENDEROVICH[1], JIACHEN ZHANG[2], ELDAN COHEN[2], AND J. CHRISTOPHER BECK[2]
[1]School of Information Technology, York University, Toronto, ON M3J 1P3, Canada
[2]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON M5S 1A1, Canada

Corresponding author: Arik Senderovich (sariks@yorku.ca)

The authors would like to thank Fujitsu Ltd. and Fujitsu Consulting (Canada) Inc. for providing financial support and access to Digital Annealer at the University of Toronto. The research was partially supported by the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT A key aspect of the design of a wind farm is the wind farm layout optimization (WFLO) problem: given a wind farm site and information about the wind patterns, the problem is to decide the location of individual wind turbines to maximize energy production subject to proximity restrictions and wake-based interference between turbines. Given the pairwise wake interactions, it is natural to model the energy objective as a quadratic function as, indeed, has been done in some existing optimization approaches. However, state-of-the-art solutions often trade-off between speed in producing designs and quality in terms of finding and proving optimal solutions. In this work, we aim to find a balanced approach to obtain WFLO solutions quickly for interactive design *and* solve the problem to optimality when quality is more important. To this end, we exploit recent advances in optimization hardware and software that target quadratic constraints: commercial mixed integer linear solvers have been extended to address some quadratic problems and nascent specialized hardware, including quantum computing systems, have focused on solving quadratic unconstrained binary optimization (QUBO) problems. We introduce two novel quadratic programming models for WFLO: a quadratic constrained optimization problem (QCOP) with binary decision variables and a QUBO. A thorough numerical evaluation using a commercial solver and specialized QUBO hardware show that our quadratic framework achieves fast, high-quality solutions that improve the state of the art and strike a balance between speed and quality. In particular, the QUBO model delivers high quality solutions in a few seconds while the QCOP model can be used to find better solutions and provide quality guarantees over a longer run-time.

INDEX TERMS Quadratic unconstrained binary optimization (QUBO), quadratic programming (QP), digital annealing, wind farm layout optimization (WFLO).

I. INTRODUCTION

The goal in the wind farm layout optimization (WFLO) problem [1] is to locate wind turbines within a predefined area to maximize the total power harvested from the wind stream. While physical considerations based on the diameter of the blades prevent turbines being placed too close to each other, turbine location decisions influence the total power production due to interference effects (i.e., wakes) generated by upstream turbines. These inter-turbine effects are best captured by Jensen's wake model [2] when superimposed using a sum-of-squares expression [3]. However, this wake modeling yields intractable optimization problems that have been approximated using either quadratic or linear functions [4]. An alternative formulation of WFLO relies on a less accurate wake model, using linear superpositioning to capture multiple wakes [5]. The latter can be captured using quadratic constraints and objective functions. However, exact optimization approaches that have been proposed for the alternative WFLO formulation [3], [5]–[9] represent the quadratic relations using approximate linear models, as solving quadratic programming problems is computationally challenging compared to linear models.

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Anvari-Moghaddam.

In recent years, there has been a growing need to quickly find high quality (albeit not always optimal) WFLO solutions [10]. The focus on quickly solving the problem was primarily motivated by the fact that modern farms cover large areas and include hundreds, and sometimes even thousands, of turbines [11]. The layout design process is iterative, computationally expensive, and controlled by various stakeholders. For each iteration, designers must either alter an incremental layout or propose a new design which they have generated by incorporating new constraints and parameter values. Hence, one wishes to solve WFLO interactively at design time, while changing many of the input parameter values and performing a sensitivity analysis to test the impact of those changes [12].

In this work, we introduce two novel quadratic WFLO models with the goal of combining efficiency, physical farm representation, and optimality. The first model represents WFLO as a quadratic constrained optimization problem (QCOP). The QCOP can be solved using state-of-the-art mathematical software that has been recently extended for such problems, e.g., Gurobi [13]. Our second model is a quadratic unconstrained binary optimization (QUBO) representation of the WFLO problem. Such a formulation enables the use of nascent specialized optimization hardware tailored to solve QUBOs: we use Fujitsu's Digital Annealer (DA) [14]. By employing the two types of advances (software and hardware) for solving quadratic problems, we are able to quickly find high quality solutions in terms of the total energy. Moreover, our approach is based on a physical model of the wind farm, taking into account closed-form calculations of the total energy and the wake effects. These two models together achieve a balance between efficiency and effectiveness of the resulting design, in a model-based fashion.

The main contributions of this paper are as follows:

1) We propose a novel quadratic modeling framework for WFLO that encompasses both constrained and unconstrained quadratic optimization models.
2) We prove that the QCOP variation of WFLO is $\mathcal{NP}$-hard.
3) We show through numerical experiments that, when solved with optimization software (Gurobi), the QCOP model results in state-of-the-art performance in terms of finding solutions with guaranteed bounds on solution quality, including finding and proving optimal solutions for a subset of the tested problem instances.
4) We show through numerical experiments that, when solved with specialized hardware, the QUBO model finds solutions competitive with the state of the art in one second, representing a computational speed-up of two to three orders of magnitude compared to software based methods. Such a short runtime enables an interactive wind farm design cycle.

Our experiments are focused on the performance of the two approaches compared to existing optimization methods. To this end, we experiment with 12 standard WFLO benchmark problems (c.f. [4]) and show that solving quadratic WFLO models using Gurobi and the DA achieves state-of-the-art performance.

The remainder of this paper is structured as follows. Section II presents an abstraction of the wind farm setting leading to WFLO, while Section III provides an overview of the related work on exact and approximate optimization solutions to the layout problem. The main contribution, namely the two quadratic models, is introduced in Section IV and Section V demonstrates the value of our approach via a thorough numerical evaluation. In Section VI, we discuss the results and their practical implications. Lastly, Section VII provides concluding remarks and directions for future work.

II. PROBLEM SETTING

In this section, we present the physical problem setting of laying out a wind farm based on the common notation and description of Zhang *et al.* [3].

A. WIND FARM AS A GRID

We consider a square wind farm that we model as an $n \times n$ grid with n being the number of cells on each axis. The cells have an area of c^2 (in square meters), i.e., if $c = 100$ meters and $n = 20$, the farm area will be 2 km $\times$ 2 km. We assume that the terrain is flat and that we have no constraints related to the terrain. Furthermore, we do not account for the influence of noise on the wind farm's surroundings. The wind farm literature treats the more complex farm models that capture non-flat terrain [6], [15], multi-typed turbines [16], noise effects [3], [17], [18], and complex objectives (including turbine installation and maintenance costs) [9], [19]. In this work, we choose to focus on a novel approach for solving WFLO, and hence consider a more abstract wind farm model. Extension of our framework to include such additional problem characteristics is left as future research.

As is common, we assume that the number of identical turbines to be placed is known in advance and denoted by m. To place these m turbines, we must consider two types of interference effect: *proximity constraints* and *wakes*. In what follows, we explain the two effects using a toy example with an 8×8 grid (see Figure 1).

B. PROXIMITY CONSTRAINTS

Turbines must be placed at least *five rotor diameters* apart. Depending on the cell size, we formulate the corresponding constraint as part of the optimization model. For example, if a turbine is placed in the center of a cell, a cell size of $c = 100$ meters and a rotor diameter of 40 meters, requires that two turbines are not placed in adjacent cells. In Figure 1, the red area around the turbine represents the forbidden cells. Conversely, if $c = 200$ and turbines are placed in the center of the cell, no proximity constraints are required as the required separation is enforced by the granularity of the spatial discretization.

C. WAKES

Turbines influence each other through *wake* interference effects that reduce the effective power produced by a turbine

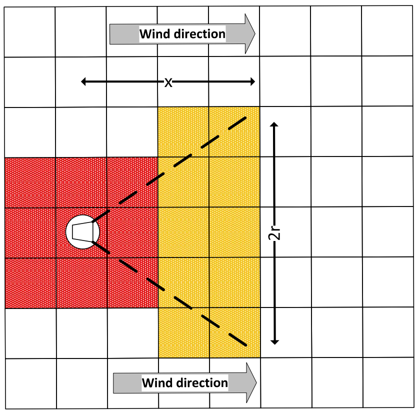


FIGURE 1. **Wind farm model: proximity constraints, wakes, wind direction.**

due to upstream turbines that change the airflow dynamics [2], [20]. In Figure 1, the wind is assumed to be blowing from left to right. The wake effect due to placing the turbine expands as shown in the diagram. If a turbine is placed in the orange area, its effective power is reduced due to the upstream turbine. Moreover, a given turbine may be placed downstream from multiple turbines and hence multiple wakes must be taken into account for each location. The reduction of power due to the wake effect is determined by three parameters: (1) the distance between the upstream turbine and the cell for which we want to compute the effect (denoted by x in Figure 1), (2) the radius of the cone's opening at distance x (denoted by r in Figure 1), and (3) the combination of several wind regimes where a wind regime consists of the wind's free speed and direction. The wake effect can be pre-computed for each pair of cells since we know the distance (x), the turbine specifications and terrain conditions (that jointly dictate r), and the probabilistic behavior of wind regimes [3].

D. EFFECTIVE POWER CALCULATIONS IN THE PRESENCE OF WAKES

Let $\mathcal{D}$ be the set of wind regimes. The probability of wind regime $d \in \mathcal{D}$ is given by p_d with $\sum_{d\in\mathcal{D}} p_d = 1$. In the experimental section (Section V), we provide one example of a probability distribution function over 36 wind regimes (Figure 2). In that example, one regime $d = (10°, 12 \text{ km/h})$ corresponds to a north-easterly wind blowing at a free velocity of 12 km/h. The probability of this regime is 0.008.

Let u_{id} be the wind velocity at turbine $i = 1, \ldots, m$ for wind regime $d \in \mathcal{D}$. Given the interference of the wakes, the velocity of turbine i is not equivalent to the free wind speed of d. Rather, u_{id} can be computed using the following equation [3]:

$$u_{id} = u_{id\infty}\left[1 - \sqrt{\sum_{j\in\mathcal{U}_{id}} \left(1 - \frac{u_{ijd}}{u_{id\infty}}\right)^2}\,\right], \tag{1}$$

with $u_{id\infty}$ being the free wind speed (the second component of regime d), $\mathcal{U}_{id}$ corresponding to the set of turbines that are upstream to i under wind regime d, and u_{ijd} being the wake-reduced speed at i due to upstream turbine j. The reduced speed, u_{ijd}, is computed using the distance x between the two turbines (i, j) and turbine and terrain specifications that yield the cone opening radius, r. The set of upstream turbines, $\mathcal{U}_{id}$ is computed by rotating the grid wrt to the current wind regime $d \in \mathcal{D}$ and identifying the turbines, j, located upstream of turbine i given d. The expression in Eq. (1) is referred to as the *sum-of-squares* (SS) model for the total expected power in presence of wakes [3]. We can now write the sum-of-squares (SS) expected power of a wind farm with m turbines:

$$E_{SS} = \sum_{i=1}^{m} \sum_{d\in\mathcal{D}} \frac{1}{3} u_{id}^3 p_d. \tag{2}$$

Eq. (2) is considered the most accurate analytical total power expression that accounts for multiple wakes [2]. However, the expression is challenging for exact mathematical optimization techniques due to its complexity (i.e., the cube of a square root).

An alternative form of the total expected power in the presence of multiple wakes is the linear superposition (LS) expression:

$$E_{LS} = \sum_{i=1}^{m} \sum_{d\in\mathcal{D}} \left(\frac{1}{3} u_{id\infty}^3 - \sum_{j\in\mathcal{U}_{id}} \frac{1}{3}(u_{id\infty}^3 - u_{ijd}^3)\right). \tag{3}$$

Note that E_{LS} has the same parameters as E_{SS} but differs in their relationship. The LS model is less accurate compared to SS [3], yet as we will show in Section IV, it results in a quadratic objective function in our WFLO formulations. Such functions have been the subject of recent research advances [21], [22], motivating the contributions here.

III. OPTIMIZATION METHODS FOR WFLO

The allocation of turbines in a grid-like wind farm was first considered as an optimization problem by Mosetti *et al.* [1]. Their method employs a genetic algorithm [23], an approximate technique with no guarantees of solution quality but with the goal of quickly finding a high-quality turbine placement.

In subsequent work [3], [4], the WFLO problem was solved using *exact* optimization methodologies that guarantee that the returned solution is optimal with respect to the specified objective or is probably within a given bound of optimal. The exact methods were shown to yield higher energy values, yet they often suffered from high computational cost and long run-times. In practice, it is often desirable to strike a balance between optimality and computational requirements: fast sub-optimal solutions are useful when solving the wind farm design problem numerous times (e.g., when considering different numbers of turbines and various allowed locations), while optimal solutions yield the best possible placements (admitted by the mathematical model) when longer run-times are available.

In what follows, we provide a literature review that covers both exact and approximate methods for WFLO. We subsequently relate our work to both approaches via our proposed quadratic optimization framework for WFLO.[1]

A. EXACT OPTIMIZATION METHODS

The WFLO problem can be solved to optimality using techniques from operations research (e.g., mixed-integer linear programming (MILP), quadratic programming (QP), and constraint programming (CP)) [3]–[5]. These methods are *exact*, in the sense that they guarantee that the returned solution is indeed the best solution one can obtain for the given model of the problem. Donovan was the first to solve WFLO using a MILP approach based on the LS energy expression [5]. Turner *et al.* [4] approximated the SS energy expression using a quadratic and a linear approximation. The two resulting methods performed well when compared to existing approximate and exact techniques. Another non-linear optimization approach [24] relaxed the binary placement variables to continuous values in [0, 1] and approximated the constraints such that the resulting variables yield an integer placement. Their approach provides another approximation to the WFLO problem considered by Turner *et al.* [4].

Zhang *et al.* [3] developed a constraint programming (CP) approach to the problem variation presented above. CP is an optimization technique that has grown from the AI literature that does not make the assumptions on the functional form of the constraints or objective [25]. The authors compared existing MILP and QP models to their approach for directly optimizing the SS objective and showed that, while promising for smaller problems, applying CP was computationally intractable for the larger standard WFLO benchmarks.

In our work, we build upon the approach in Donovan [5], yet instead of using a linear version of the LS objective, we model WFLO using a quadratic representation. Unlike Turner *et al.* [4] and Ulku and Alabas-Uslu [24], our model is not an approximation of the SS objective, but rather an exact representation of the LS objective. That is, while the previous work has approximated a more accurate physical model, we are exactly representing the less accurate physical model: the approximation arises at a different stage of the modeling pipeline.

While the main advantage of exact techniques is that they provide guarantees on solution quality, they may take a very long time to find an optimal solution. In fact, Section IV shows formally that WFLO is computationally hard, a result that is not surprising but that has apparently not explicitly appeared in the WFLO literature. Therefore, when one considers solving multiple wind farm design problems, if a fast solution is of essence, one may turn to approximate optimization techniques.

[1] We review only literature that considers the same wind farm model as the one presented at the beginning of this section. For a broader literature review see Zhang *et al.* [3].

B. APPROXIMATE METHODS

In contrast to exact methods, approximation algorithms often provide quick and, hopefully, sufficiently high-quality solutions. Several works have solved WFLO using evolutionary (genetic) approaches [1], [26], [27]. These methods do not guarantee termination (i.e., they may run forever without achieving a quality criteria), and even when they do terminate they do not guarantee optimal solutions. Hence, one must often introduce non-quality related stopping criteria, compromising the quality of the obtained solution [23]. Furthermore, it is awkward to introduce constraints into genetic methods (see e.g., [9]), leading to potentially costly feasibility checks and problem-specific repair operations for every solution found.

The first attempt to address these limitations was made using a local search approach [28]. The approach does not guarantee optimality, yet it circumvents the termination and feasibility checking issues of the evolutionary methods. The local search attempts to apply either move, remove, or add actions given an incumbent feasible solution. When a turbine is moved, removed, or added, the new solution is evaluated. The procedure terminates when a stopping criteria is reached. The main limitation of this approach is that it cannot perform an action that leads to an infeasible state and, therefore, often quickly converges to a sub-optimal solution (a local optimum) that can potentially be much lower quality than the globally optimal solution [29].

To overcome this limitation, Rivas *et al.* [29] used simulated annealing (SA), a neighborhood search method that creates a connected solution space that allows the algorithm to escape local optima. The SA method was shown to be superior to the approach of Ozturk & Norman [28] and to the genetic method proposed in Grady *et al.* [27]. A drawback of the SA approach is its ad-hoc nature. The various components of the algorithm must be tailored to the problem at hand and, as a result, small adjustments of the WFLO setting (e.g., adding noise considerations), would require major changes to the SA implementation. Additional methods and experimental comparisons between various approximate algorithms are available in Samorani [30].

In our work, we also use an annealing based approach (similar to Rivas *et al.* [29]). However, our methodology is generic since it is based on a declarative problem model and on the ability of specialized hardware to solve quadratic problems.

Recently, several efficient WFLO solutions based on developmental models were introduced [10], [11]. These works do not represent the actual underlying physical model of the system, and while providing fast solutions to WFLO, they do not attempt to optimize on the actual objective function (that is, the total energy that a farm produced) nor capture the complexities of the underlying problem. For example, one approach [11] models turbine interference using a parametric probabilistic model (assuming a Weibull distribution). In our quadratic models, we explicitly represent the energy term, and the physical dependencies between the turbines (i.e., wakes and proximity constraints). Furthermore, our models optimize an empirically established energy objective, the linear superposition expression in Eq. (3).

IV. QUADRATIC MODELS FOR WFLO

In this section, we present our main methodological contributions, namely our framework of quadratic (constrained and unconstrained) optimization for solving WFLO, and the proof that the computational complexity of the constrained problem is $\mathcal{NP}$-hard.

A benchmark study comparing state-of-the-art exact approaches [3] to approximate methods found that the two perform in a comparable fashion, without large differences between the attained energy [31]. Exact methods tend to perform slightly better in smaller WFLO benchmarks, while simulated annealing methods dominated for larger standard WFLO benchmarks. These results led to the conclusion that both types of methods (exact and approximate) should be considered. Therefore, we introduce a unified paradigm, namely quadratic programming, that enables both exact and approximate solutions to WFLO. Further, formulating WFLO using quadratic programming enables us to employ recent advanced exact software solutions (e.g., Gurobi optimizer) and approximate hardware developments (e.g., Fujitsu's Digital Annealer).

We consider two quadratic optimization problems that represent WFLO: a quadratic constrained optimization problem (QCOP), which provides exact solutions and a quadratic unconstrained binary optimization (QUBO) problem, which results in approximate solutions. The former represents WFLO characteristics (proximity and number of turbines) as hard constraints, while the latter considers these characteristics as soft constraints such that the objective function is penalized when these constraints are violated.

We start by presenting the inputs and decision variables of our optimization approaches. Subsequently, we introduce the two quadratic formulations of WFLO and prove its computational complexity. Lastly, we discuss the details of existing software and hardware solutions that can be used to solve the two formulations.

A. WFLO INPUTS AND DECISION VARIABLES

Given the wind farm problem presented in Section III, let x_i be a binary decision variable that represents whether a turbine is positioned at location $i \in \mathcal{N}$ with $\mathcal{N}$ being the set of $k = n^2$ possible turbine locations (cells in the grid). We write x as shorthand for $x = (x_1, \dots, x_k)$. We denote by $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ the set of location pairs that cannot simultaneously host turbines due to proximity constraints. The set $\mathcal{E}$ can be pre-computed based on problem specifications.

Let u_{id} and $u_{id\infty}$ be the wind speeds at turbine location $i \in \mathcal{N}$ for wind regime $d \in \mathcal{D}$ with and without interference from other turbines due to wake effects, respectively. Note that here we refer to i as a potential turbine location and not the ith turbine as we did in Section III. Further, we denote by $\mathcal{U}_{id} \subseteq \mathcal{N}$ the set of upstream turbine locations for wind regime d given that a turbine is placed at cell i (i.e., turbines placed in these locations will influence a turbine placed in i). Finally, we denote by u_{ijd} the wind speed at location j due to a single wake from upstream turbine at location i with $j \in \mathcal{U}_{id}$. The following function represents the total expected energy of a wind farm given placement decisions x:

$$f(x) = \sum_{i \in \mathcal{N}} \sum_{d \in \mathcal{D}} p_d \Big(\frac{1}{3}\, u^3_{id,\infty} x_i - \sum_{j \in \mathcal{U}_{id}} \frac{1}{3} (u^3_{id,\infty} - u^3_{ijd}) x_i x_j \Big). \tag{4}$$

The objective function corresponds to the linear superposition (LS) expression (c.f., Eq. 3 in Section III). We are now ready to provide the two quadratic programs to represent WFLO.

B. WFLO AS A QUADRATIC CONSTRAINED OPTIMIZATION PROBLEM

1) THE QC-LS WFLO MODEL

To maximize $f(x)$ and satisfy the total number of turbines and proximity constraints we write the following quadratic constrained optimization problem (QCOP):

$$\max_{x_i} \sum_{i \in \mathcal{N}} \sum_{d \in \mathcal{D}} p_d \Big(\frac{1}{3}\, u^3_{id,\infty} x_i - \sum_{j \in \mathcal{U}_{id}} \frac{1}{3} (u^3_{id,\infty} - u^3_{ijd}) x_i x_j \Big)$$

$$s.t.: \quad \sum_{i \in \mathcal{N}} x_i = m, \tag{5}$$

$$x_i + x_j \leq 1, \forall (i,j) \in \mathcal{E}, \tag{6}$$

$$x_i \in \{0, 1\}, \forall i \in \mathcal{N}. \tag{7}$$

Constraints 5 and 6 ensure that exactly m turbines are placed on the grid and enforce proximity constraints between the turbines, respectively. We shall refer to the QCOP formulation of WFLO as **QC-LS**.

QC-LS can be solved using an exact optimization solver, e.g., Gurobi [13]. Linear versions of the QCOP (**QC-LS**) have been previously proposed and solved using the corresponding exact solvers [3], [5].

2) THE COMPUTATIONAL COMPLEXITY OF QC-LS

Despite the existence of mathematical models for WFLO in the literature [3], [5] that are expressive enough to represent $\mathcal{NP}$-hard problems, we are unaware of an explicit treatment of the computational complexity of WFLO. We therefore provide a straightforward proof of the computational complexity of **QC-LS**.

Theorem 1: The computational complexity of **QC-LS** defined in (5)-(7) is $\mathcal{NP}$-hard.

Proof 1: We prove by showing that a special case of the **QC-LS** (5)-(7) is $\mathcal{NP}$-hard. Suppose that $\mathcal{E} = \emptyset$, i.e., the grid is coarse-grained enough to ignore proximity constraints. Then, the constraints in Eq. (6) are dropped and we get the heaviest k-subgraph problem, which is a known $\mathcal{NP}$-hard problem [32]. Since a special case of **QC-LS** is $\mathcal{NP}$-hard, **QC-LS** is at least $\mathcal{NP}$-hard. On the other hand, **QC-LS** is formulated using integer programming (Problem 5), which means that it is at most $\mathcal{NP}$-hard. Therefore, we get that **QC-LS** is $\mathcal{NP}$-hard.

3) QCOP SOLVING METHODS

QC-LS is a strictly convex integer quadratically constrained optimization problem (IQCOP), which is notorious for its computational complexity [33]. The common exact methods for solving IQCOPs are tree search [34] algorithms with a variety of sophisticated techniques to limit the search necessary to find and prove optimality. In operations research, the generic exact approach for solving IQCOPs is the branch-and-cut based mixed integer programming (MIP) [35].

Specifically, the branch-and-cut approach is mainly composed of branch-and-bound (BB) and cutting-planes methods [36]. The BB method is a systematic enumeration framework [37], where the bounding part computes lower bounds by solving the continuous relaxations of the original problem, while upper bounds are provided by heuristics and feasible solutions encountered during the search. The branching part intelligently splits the search space into smaller spaces according to bounds and branching heuristics. The cutting-plane method iteratively refines the search space by introducing linear inequalities, known as cuts, to strengthen the lower bound in the BB method [38]. An optimal solution is proved when the lower and upper bounds meet.

Commercial MIP solvers have seen extensive development over the past 20 years with hardware-independent algorithmic improvements delivering orders of magnitude improved performance [39]. More recently the development of such solvers has focused on representing and solving quadratic problems such as **QC-LS** [40].

C. WFLO AS A QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION PROBLEM

1) THE QUBO WFLO MODEL

Quadratic unconstrained binary optimization problems (QUBOs), as their name implies, do not allow the direct representation of hard constraints. Instead, problem constraints must be represented as part of the objective function using penalty terms.

Let $\lambda = (\lambda_1, \lambda_2)$ be a vector of constraint weights with $\lambda_i > 0$. The equivalent QUBO formulation of the QCOP in Eq. (5) is given by,

$$\max_x f(x) - \lambda_1 (\sum_{i \in \mathcal{N}} x_i - m)^2 - \lambda_2 \sum_{(i,j) \in \mathcal{E}} x_i x_j, \tag{8}$$

with the term $\lambda_1(\sum_{i\in\mathcal{N}} x_i - m)^2$ penalizing solutions that violate the exactly m turbines constraint and the term $\lambda_2 \sum_{(i,j)\in\mathcal{E}} x_i x_j$ penalizing solutions that violate the proximity constraints. The two penalty terms λ_1 and λ_2 must be large enough to ensure that the solutions are indeed feasible wrt the problem constraints. We shall refer to the QUBO representation of WFLO as **QU-LS**.

2) SOLVING QUBO VIA SPECIALIZED HARDWARE

Recent years have seen significant advancement in the use of specialized hardware platforms, such as adiabatic and gate-based quantum computers, digital/CMOS annealers, and neuromorphic computers [41], to solve hard combinatorial optimization problems. A large number of these platforms support the optimization of quadratic functions over binary variables by using Ising models [42], an equivalent representation to QUBO [22], as their mathematical abstraction.

In this work, we use the Fujitsu Digital Annealer (DA), a recent computer architecture designed for solving QUBO problems [43]. The third generation DA (DA3), a hybrid system of hardware and software, is capable of representing QUBOs with up to 100000 variables. For our DA environment,[2] the coefficients for the quadratic terms range from -2^{62} to 2^{62} and those for the linear terms range from -2^{73} to 2^{73} [44]. The algorithm used by the DA is based on simulated annealing [45], however it takes advantage of the massive parallelization provided by the custom CMOS hardware and uses a dynamic offset mechanism to escape local minima [14].

Specifically, DA searches in a large solution space through a Markov-Chain Monte Carlo (MCMC) process by evaluating a 1-bit flip neighborhood (all states with Hamming distance of 1 of a QUBO state) at each step [46]. Through parallelism, DA compares all 1-bit flip neighbors of the current QUBO state and stochastically selects one that reduces the QUBO cost, if such a state exists, or a worse solution, otherwise. DA has a parallel tempering mechanism that uses multiple replicas (MCMC processes) at different temperatures in parallel and swaps temperatures for replicas to better escape local optima and locate high-quality solutions [47]. The latest version of the architecture supports a dedicated bit flip mechanism, over a subset of variables belonging to one-hot equivalent constraints.

The Fujitsu Digital Annealer has been previously applied to a variety of problems in different research areas such as machine learning [48], [49] and communication [50], [51].

Unlike standard exact optimization solvers that run until obtaining an optimal solution, the DA runs until reaching a time limit and returns the best found solution. In our experiments, we analyze the quality of solutions for different time limits.

V. EVALUATION

In this section, we present a numerical evaluation of the two quadratic models based on 12 standard WFLO instances. Our main results are:

- For given time limit, the Digital Annealer (DA) achieves a solution quality as good or as better than all tested state-of-the-art exact methods on 8 of 12 instances.
- This performance is achieved for the **QU-LS** model with a runtime that is two to three orders of magnitude shorter

[2]All experiments were conducted on the Digital Annealer environment prepared exclusively for the research at the University of Toronto.

than QC-LS on state-of-the-art commercial solvers for hard WFLO instances.

- Given sufficient runtime, solving the QC-LS model with commercial software yields strong results on most of the instances. Compared to all tested previous exact models (such as QC-SS), QC-LS is able to find equal or better solutions on 6 of 12 benchmark problems and for 3 of the 6 hard ones.

We start by outlining the experimental design of our evaluation, followed by our main results, and conclude the section with a discussion of the results and their practical implications.

A. EXPERIMENTAL DESIGN

a: STANDARD WFLO INSTANCES

To test and benchmark our models, we used 12 synthetically generated, benchmark WFLO instances from the literature [1], [3], [27] that we denote with the set $\mathcal{W}$. These benchmarks are standard instances used in many optimization studies of WFLO, as they represent a matching input into the abstract problem formulation. The 12 benchmarks result from varying two wind settings (1 and 36 wind directions), two grid sizes (10×10 with cell size 200 meters and 20×20 with cell size 100 meters), and three turbine cardinalities ($m \in \{20, 30, 40\}$). For the 36 wind directions setting, the velocities come from the probability distribution presented in Figure 2. The x-axis corresponds to possible wind directions in units of 10 degrees, e.g., the value 5 corresponds to 50 degrees. For each wind direction, the wind can blow with three velocities, namely 7, 12, and 17 meters per second. Each of these velocities has a probability of occurring at a given angle. These probabilities correspond to the y-axis of the figure. For example, for 50 degrees, the probability of 17 meters per second wind is just above 0.01, while the probability of velocity being 12 meters per second is around 0.008. Lastly, the probability of having a velocity of 8 meters per second is below 0.005. The sum of all probabilities per velocity across the various angles is 1. Instances with 36 wind regimes are considered computationally more challenging [3].

Additional parameters that were used to calculate the velocities ($u_{id\infty}, u_{ijd}$) include: turbine height (60 meters), ground roughness (0.3 meters), turbine rotor radius (20 meters), and a wind farm of size 2km $\times$ 2km.[3]

b: OPTIMIZATION BENCHMARKS

We compared our two models, QC-LS and QU-LS, against two state-of-the-art exact optimization methods: the integer linear programming (denoted ILP-LS) approach [3], [5] and the model that uses a quadratic approximation (QC-SS) [4]. The ILP model is a standard, exact linearization of the QC-LS model. However, the ILP is solved using linear optimization approaches, while QC-LS is solved using quadratic programming methods as mentioned in Section IV-B3. Hence we can expect a difference in the performance of optimization software when solving the two problems.

The existing constraint programming approach was reported to perform worse than the ILP model [3] and so was omitted from our the experiments. We used state-of-the-art commercial software, Gurobi v9.1.1 [13], to solve QC-LS, ILP-LS, and QC-SS. We shall denote the problems using their names and the solvers used as follows: QC-LS(GRB) (GRB for Gurobi), ILP-LS(GRB), and QC-SS(GRB). In addition, we used both Fujitsu's Digital Annealer and Gurobi to solve QU-LS: we shall refer to the former as QU-LS(DA) and to the latter as QU-LS(GRB).

c: EXPERIMENTAL PROCEDURE

For exact methods (QC-SS, ILP-LS, QC-LS, and QU-LS), we ran Gurobi with a time limit of 3600 seconds. We collected all feasible solutions found until that point in time. The exact models were solved using Gurobi's integer programming and quadratic programming solvers on a Windows PC with Intel(R) Core(TM) i7-8700K CPU @3.20GHz with 16 GB RAM.

For the DA, we used a single configuration that was found to work best across the different instances. The DA was run for increasingly large time limits ranging from 1 second to 10 seconds, at which point we no longer observe an improvement in performance. The penalty terms λ_1 and λ_2 (see Eq. (8)) are selected such that $\lambda_1 = \lambda_2$ and their magnitudes are tuned by DA3 automatically.

d: PERFORMANCE MEASUREMENT

We measured the total expected energy of a solution using the SS expression (see Eq. (1)). The only method that directly optimizes SS is QC-SS(GRB). Hence, for the other methods, we converted LS-based solutions into the SS expression using Eq. (1) as proposed in Zhang *et al.* [3]. Specifically, let $x = (x_1, \ldots, x_k), k = n^2$ be a turbine placement solution obtained by an LS-based method. Then, the SS energy expression can be written as,

$$\sum_{i \in \mathcal{N}} \sum_{d \in \mathcal{D}} \frac{1}{3} x_i \left(u_{id,\infty} \left[1 - \sqrt{\sum_{j \in \mathcal{U}_{id}} x_j \left(1 - \frac{u_{ijd}}{u_{id,\infty}} \right)} \right] \right)^3 .$$

Next, we tested the performance of our approach as function of solution time. We compared the average performance of all WFLO methods on all 12 WFLO instances. Since energy levels are not directly comparable between the different instances (due to varying number of turbines and wind farm specifications) we used the *mean relative error* (MRE) measure with respect to the best solution obtained per instance.

Recall that $\mathcal{W}$ is the set of our 12 standard WFLO instances. Let b_w be the best energy solution (in terms of SS) obtained for an instance $w \in \mathcal{W}$ across the different methods. Then, for every point in time $t \in \{0, \ldots, 3600\}$ (measured in seconds), we can compute the relative error of a given approach per instance w. Let $\mathcal{A}$ be the set of solution

[3]The implementation of construction of the 12 standard WFLO instances can be found in https://bit.ly/2YIjTl1

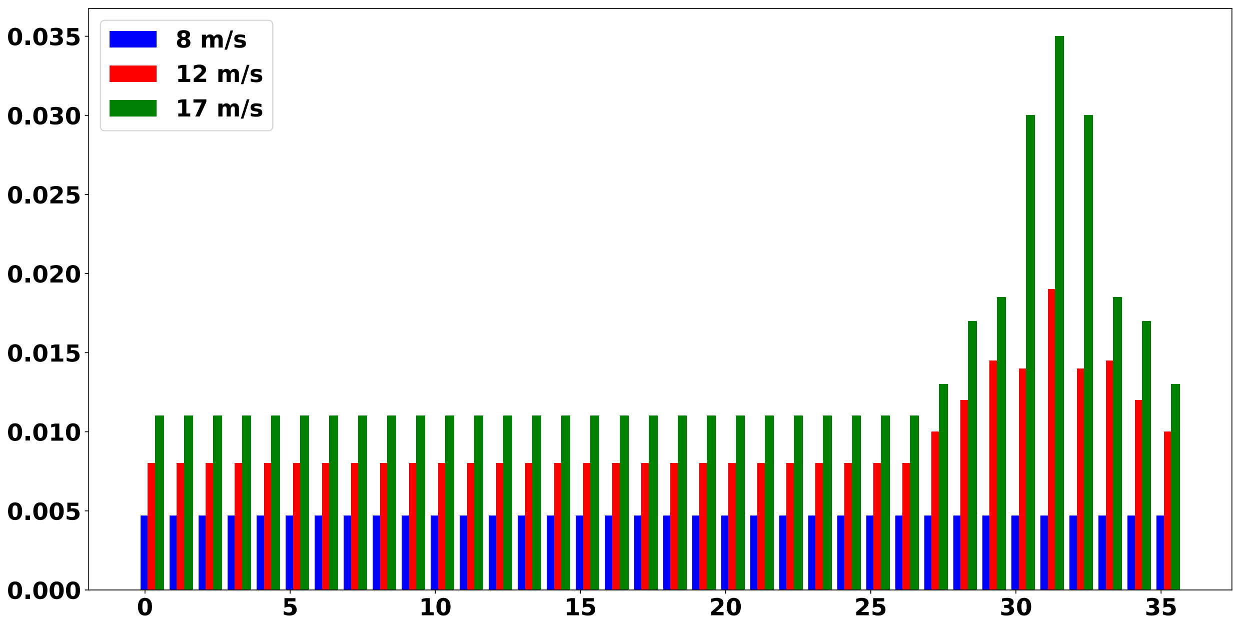


FIGURE 2. Wind regimes: x-axis is the angle (times 10 degrees), y-axis is the probability for wind regime, and color corresponds to free wind speed for the different wind directions.

approaches, i.e., $\mathcal{A} = \{$*QU-LS(GRB)*, *QC-LS(GRB)*, *QU-LS(DA)*, *ILP-LS(GRB)*, *QC-SS(GRB)*.$\}$ Further, denote by $b_{w,t,a}$ the best solution attained before time t by approach a for instance w. Then, the relative error at time t for approach a on instance w is given by

$$RE(w, t, a) = \frac{b_w - b_{w,t,a}}{b_w}. \tag{9}$$

Note that the expression is always non-negative, since we are solving a maximization problem. We can now average over the different instances at time t. The average performance of approach a at time t, $MRE(t, a)$ can be computed as,

$$MRE(t, a) = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} RE(w, t, a). \tag{10}$$

We present $MRE(t, a)$ for each of the approaches to demonstrate the speed of convergence of the proposed LS-based methods.

The last measure that we introduce is the speed-up factor $S(a_1, a_2, t, w)$. The measure compares the time that it takes approach $a_1 \in \mathcal{A}$ to achieve the same energy level as the energy achieved by approach $a_2 \in \mathcal{A}$ within t seconds when solving instance w (energy levels are measured in terms of the sum-of-squares energy expression). Formally, let $\phi(a_1, a_2, t, w)$ be the set of all times where a_1 achieves similar or better energy level as a_2 at time t,

$$\phi(a_1, a_2, t, w) = \{t' : E_{SS}(a_1, t', w) \leq E_{SS}(a_2, t, w)\},$$

with $E_{SS}(a, t, w)$ being the SS energy value under approach a for instance w at time t. The speed-up factor within t seconds is then defined as follows

$$S(a_1, a_2, t, w) = \frac{\min\left[\{\phi(a_1, a_2, t, w)\} \cup \{3600\}\right]}{t}. \tag{11}$$

The speed-up factor is useful to measure how many times faster (or slower) approach a_1 solves problem w compared to approach a_2. For example,

$$S(\textit{QC-LS(GRB)}, \textit{QU-LS(DA)}, 10, w_1)$$

is the ratio of the time that it takes QC-LS(GRB) to reach the same quality of solution as QU-LS(DA) achieves after 10 seconds on problem w_1. Since we limit the runtime of the exact methods by 3600 seconds, S is a lower bound on the true speed-up factor.

B. MAIN RESULTS

Table 1 contains the best energy (in kW) attained by the methods for the 12 WFLO instances after 100 seconds of runtime for the exact methods. The runtime for the DA was always less than 10 seconds.

We observe that the DA-based solution provides comparable results to state-of-the-art exact methods. For the harder instances (with 36 wind directions), the three quadratic LS-based approaches (QU-LS(DA), QU-LS(GRB), and QC-LS(GRB)) dominate the others by finding the best (or equal) solutions in all 6 instances. For easier instances, the DA-based approach (QU-LS(DA)) also dominates as it finds the best solutions in 5 out of 6 instances. Table 2 shows the same type of results as in Table 1, but now after 3600 seconds for exact methods. For the exact technique, we can see that the improvement between 100 and 3600 seconds of run-time is marginal. We also observe that the DA at 10 seconds is, therefore, still competitive with the other approaches, while QC-LS(GRB) finds the best solutions in 6 out of the 12 scenarios. Our quadratic approaches dominate all six hard scenarios (36 wind directions).

TABLE 1. Total expected power (kW) after 100 seconds for the exact techniques and by 10 seconds for QU-LS(DA).

Wind Directions	n	m	QU-LS(DA)	QU-LS(GRB)	QC-LS(GRB)	QC-SS(GRB)	ILP-LS(GRB)
WR1	10	20	**11185.41**	**11185.41**	**11185.41**	**11185.41**	**11185.41**
		30	**15742.93**	**15742.93**	15740.69	15741.81	15741.81
		40	**19265.21**	18954.28	**19265.21**	**19265.21**	**19265.21**
	20	20	**11404.80**	**11404.80**	**11404.80**	**11404.80**	**11404.80**
		30	16399.56	16602.85	16761.18	**16774.37**	16760.91
		40	**21973.80**	21854.57	21907.27	21918.85	21894.62
WR36	10	20	**19221.44**	**19221.44**	**19221.44**	19175.88	19186.58
		30	27440.08	27442.90	**27443.34**	27393.49	27406.48
		40	**35409.58**	34938.65	34888.37	34824.24	34867.31
	20	20	**19437.49**	19310.96	19391.76	19333.97	19331.84
		30	**27939.08**	27689.15	27885.49	27747.88	27859.50
		40	35547.27	35352.31	**35623.11**	35229.93	35521.68

TABLE 2. Total expected power (kW) after 3600 seconds and by 10 seconds for QU-LS(DA).

Wind Directions	n	m	QU-LS(DA)	QU-LS(GRB)	QC-LS(GRB)	QC-SS(GRB)	ILP-LS(GRB)
WR1	10	20	**11185.41**	**11185.41**	**11185.41**	**11185.41**	**11185.41**
		30	**15742.93**	**15742.93**	15740.69	15741.81	15741.81
		40	**19265.21**	18954.28	**19265.21**	**19265.21**	**19265.21**
	20	20	**11404.80**	**11404.80**	**11404.80**	**11404.80**	**11404.80**
		30	16399.56	16769.09	16761.18	**16774.37**	16770.17
		40	**21973.80**	21891.25	21919.68	21918.85	21973.69
WR36	10	20	**19221.44**	**19221.44**	**19221.44**	19175.88	19186.58
		30	27440.08	27442.90	**27443.34**	27393.49	27422.53
		40	**35409.58**	34938.65	34938.65	34824.24	34867.31
	20	20	19437.49	**19437.52**	19411.21	19388.72	19337.11
		30	**27939.08**	27812.30	27885.49	27866.93	27869.54
		40	35547.27	35571.63	**35623.11**	35264.47	35539.74

We now turn to the mean relative error over time per approach. Note that lower values indicate better performance. Figure 3 presents the MRE over the first 100 seconds, while Figure 4 corresponds to the full run (after 3600 seconds).

Our proposed quadratic models, QC-LS and QU-LS, outperform the rest of the techniques across the entire runtime time horizon that was tested. Specifically, for most instances, QC-LS(GRB) gives the best solutions at 3600 seconds, while the DA-based approach QU-LS(DA) yields the best solutions after 6 seconds. However, the good performance of QU-LS only occurs when the solver is the DA. In fact, QU-LS(GRB) trails all the other methods. Furthermore, QC-LS(GRB) converges to the best solutions faster than all other exact methods. In fact, it is inferior only to the approximate QU-LS(DA) method when the runtime is relatively small.

The results also show that on average the ILP-LS(GRB) method starts with poor solutions but improves over QC-SS(GRB) after around 2 seconds.

To demonstrate the gain in runtime from using the DA, we compare the average speed-up factor between QC-LS(GRB) and a QU-LS(DA) solutions after 1 and 10 seconds. For each $w \in \mathcal{W}$ we computed the mean $S(\mathit{QC\text{-}LS(GRB)}, \mathit{QU\text{-}LS(DA)}, t, w)$ for $t \in \{1, 10\}$. The average speed-up factors were found to be 988 and 131 for 1 and 10 seconds, respectively. Per instance, the *minimum* speedup was 1 for 1 second and 0.1 for 10 seconds: for some instances QC-LS(GRB) achieved the same quality as QU-LS(DA) in less time. The *maximum* speedup was (at least) 3600 for 1 second and (at least) 360 for 10 seconds: for those instances QC-LS(GRB) was not able to find a comparable solution to QU-LS(DA) within the 3600 time limit. As a consequence, the mean QU-LS(DA) speed-up is actually higher than computed above.

When compared to the other LS-based methods, the DA speed-up is even larger than when compared to QC-LS(GRB). At 1 (10) seconds, the mean speedup factor compared to ILP-LS(GRB) is 1430 (197) and for the QU-LS(GRB) it is 1013 (134). We compare against the LS based methods as they directly optimize on the same objective as the DA.

Table 3 presents the speedup results across all problem instances for every LS-based method (computed using the SS energy formulation). The speedup is calculated according to Eq. (11). Table 4 shows similar results for the speedup factor compared to DA at 10 seconds. The average lower bound on the speedup at 1 seconds is 988, 1430, and 1013 for QC-LS(GRB), ILP-LS(GRB), and QU-LS(GRB), respectively. For the 10-second results, the average speedup factor is at least 131, 197, and 134 for the respective methods. For instances where the speedup factor in Table 4 is 10 times less than in Table 3, the DA found the 10-second solution after 1 second.

Note that the ‘‘N/A’’ values in the table arise from our use of two different energy functions, E_{SS} and E_{LS}. For some problem instances, an exact solver for QC-LS finds and proves an optimal solution for the E_{LS} objective but, when that solution is evaluated using the E_{SS} energy function, it has a lower energy than the solution found by the DA. This phenomenon is correct because the mapping from E_{LS} to E_{SS} does not necessarily preserve the ordering of solution quality. We do not include these instances in our speedup computations.

VI. DISCUSSION AND PRACTICAL IMPLICATIONS

Our results show numerically that the quadratic programming models that we introduced in Section IV (QC-LS and QU-LS) are useful in two ways. First, the two methods are powerful approaches to solving WFLO. Specifically, QC-LS yields the best performance (after 3600 seconds). Second, the QUBO model can be solved with recent computer architecture, Fujitsu's DA, as demonstrated by the

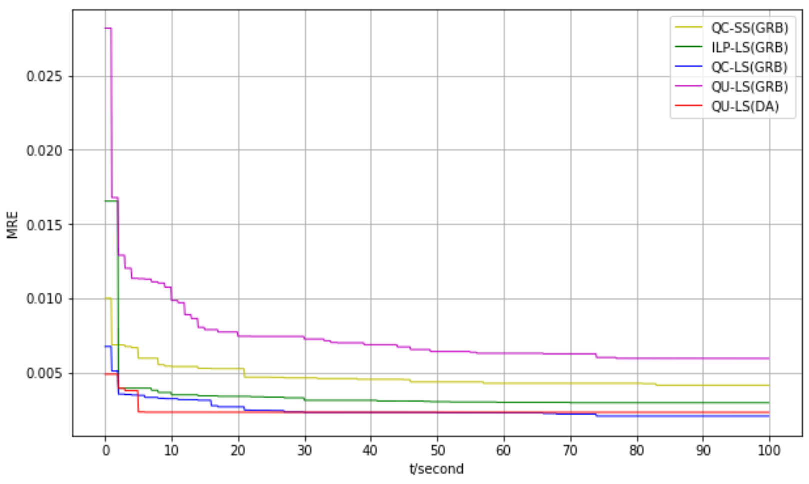


FIGURE 3. Mean relative error (wrt best known feasible solution) vs. Runtime (after 100 seconds).

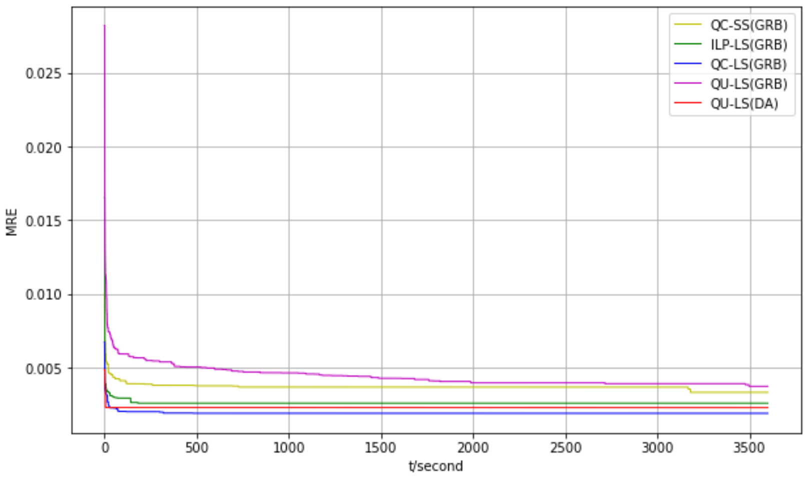


FIGURE 4. Mean relative energy (wrt best feasible solution) vs. Runtime (after 3600 seconds).

QU-LS(DA) approach. QU-LS(DA) yields the best results in 8 out of 12 problem instances (both easy instances with 1 wind direction and hard instances with 36 wind directions). In addition, the runtime performance of QU-LS(DA) is second only to the QC-LS(GRB) and outperforms all methods in terms of performance in short runtimes (a speed-up factor of three orders of magnitude when the DA runs for 10 seconds).

The ability of QU-LS(DA) to quickly obtain competitive solutions on large WFLO instances is crucial when iteratively solving multiple wind farm layout problems at design time. Moreover, it allows the flexibility to tune the various WFLO parameters (including turbine specs, number of turbines, and farm sizes) without incurring a steep computational cost that comes with (re-)solving the problem using exact methods. The quality loss due to the use of an approximate method that exploits advanced technology is negligible. This balance between runtime and total energy, highlights the contribution of the proposed quadratic framework for solving WFLO.

TABLE 3. Speedup factor per LS-based method for all instances w.r.t QU-LS(DA) at 1 second. The value 3600 implies a lower bound on the speedup factor.

Wind Directions	n	m	QC-LS	ILP-LS	QU-LS(GRB)
WR1	10	20	1	1	1
		30	N/A	N/A	1
		40	1	1	N/A
	20	20	1	1	30
		30	1	1	2
		40	3600	143	3600
WR36	10	20	1	3600	1
		30	1	1184	1
		40	1	1	1
	20	20	3600	3600	2224
		30	3600	3600	3600
		40	66	3600	1683

Finally, though we have not emphasized it here, all the approaches presented, including those from the literature, have the advantage of being represented with a declarative mathematical model. Such a representation makes problem extensions significantly easier and allows formal analysis.

TABLE 4. Speedup factor per LS-based method for all instances w.r.t QU-LS(DA) at 10 second. The value 360 implies a lower bound on the speedup factor.

Wind Directions	n	m	QC-LS	ILP-LS	QU-LS(GRB)
WR1	10	20	0.1	0.1	0.1
		30	N/A	N/A	0.1
		40	0.1	0.1	N/A
	20	20	1	1	3
		30	0.1	0.1	0.2
		40	360	14.3	360
WR36	10	20	0.1	360	0.1
		30	0.1	360	0.2
		40	360	360	360
	20	20	360	360	222.4
		30	360	360	360
		40	6.6	360	168.3

While model-based approaches are common in exact optimization such as mixed-integer programming, they are rare for heuristic techniques such as genetic algorithms and metaheuristics. Many of the state-of-the-art implementations of heuristic techniques are specialized for the particular problem being solved. A change in the problem definition (e.g., the addition of a new constraint) can, therefore, require substantial software changes. Conversely, model-based techniques can be adapted to a revised problem definition via the declarative addition of the new constraint to the model. Further, amenability to formal analysis means that substantial insight can be gained from mathematical treatments of problem classes, leading, as can be seen in mixed integer programming [39], to substantial improvements in problem solving performance.

VII. CONCLUSION

In this work, we presented a novel quadratic programming framework for solving wind farm layout optimization. The framework is embodied in two models: constrained (QC-LS) and unconstrained binary (QU-LS) quadratic programs. The former can be used to solve WFLO using advanced optimization software, while the latter can be mounted onto specialized hardware that can quickly (approximately) solve WFLO. We used the constrained quadratic formulation to prove that the computational complexity of WFLO is $\mathcal{NP}$-hard. Nonetheless, our numerical evaluation shows that when exact solvers like Gurobi [13] are coupled with our QC-LS and QU-LS models they are able to solve small WFLO instances to optimality in less than 3600 seconds. We also show that when an advanced computer architecture is used with the QUBO representation, large WFLO instances can be solved quickly with no significant degradation in the total energy compared to exact methods. This runtime improvement and state-of-the-art performance in terms of the total energy of the resulting solutions makes our quadratic framework useful for designing new wind farms, while varying multiple input parameters.

ACKNOWLEDGMENT

The authors would like to thank Fujitsu Ltd. and Fujitsu Consulting (Canada) Inc. for providing access to the Digital Annealer at the University of Toronto.

REFERENCES

[1] G. Mosetti, C. Poloni, and D. Diviacco, "Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm," *J. Wind Eng. Ind. Aerodyn.*, vol. 51, no. 1, pp. 105–116, 1994, [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167610594900809

[2] N. O. Jensen, *A Note on Wind Generator Interaction*. Roskilde, Denmark: Risø Nat. Lab., 1983.

[3] P. Zhang, D. Romero, J. C. Beck, and C. Amon, "Solving wind farm layout optimization with mixed integer programs and constraint programs," *EURO J. Comput. Optim.*, vol. 2, no. 3, pp. 195–219, Aug. 2014, doi: 10.1007/S13675-014-0024-5.

[4] S. Turner, D. Romero, P. Zhang, C. Amon, and T. Chan, "A new mathematical programming approach to optimize wind farm layouts," *Renew. Energy*, vol. 63, pp. 674–680, Mar. 2014.

[5] S. Donovan, "Wind farm optimization," in *Proc. 40th Annu. ORSNZ Conf.*, 2005, pp. 196–205.

[6] J. Y. Kuo, D. A. Romero, J. C. Beck, and C. H. Amon, "Wind farm layout optimization on complex terrains—Integrating a CFD wake model with mixed-integer programming," *Appl. Energy*, vol. 178, pp. 404–414, Sep. 2016.

[7] P. Fagerfjäll, "Optimizing wind farm layout: More bang for the buck using mixed integer linear programming," M.S. thesis, Dept. Math. Sci., Chalmers Univ. Technol., Gothenburg Univ., 2010.

[8] R. Archer, G. Nates, S. Donovan, and H. Waterer, "Wind turbine interference in a wind farm layout optimization mixed integer linear programming model," *Wind Eng.*, vol. 35, no. 2, pp. 165–175, 2011.

[9] S. Y. D. Sorkhabi, D. A. Romero, J. C. Beck, and C. H. Amon, "Constrained multi-objective wind farm layout optimization: Novel constraint handling approach based on constraint programming," *Renew. Energy*, vol. 126, pp. 341–353, Oct. 2018.

[10] D. Wilson, S. Cussat-Blanc, K. Veeramachaneni, U.-M. O'Reilly, and H. Luga, "A continuous developmental model for wind farm layout optimization," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2014, pp. 745–752.

[11] D. Wilson, E. Awa, S. Cussat-Blanc, K. Veeramachaneni, and U.-M. O'Reilly, "On learning to generate wind farm layouts," in *Proc. 15th Annu. Conf. Genetic Evol. Comput. Conf. (GECCO)*, 2013, pp. 767–774.

[12] D. Wilson, S. Rodrigues, C. Segura, I. Loshchilov, F. Hutter, G. L. Buenfil, and A. Kheiri, "Evolutionary computation for wind farm layout optimization," *Renew. Energy*, vol. 126, pp. 681–691, Oct. 2018.

[13] Gurobi Optimization. (2021). *Gurobi Optimizer reference Manual*. [Online]. Available: http://www.gurobi.com

[14] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. Katzgrabeer, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019.

[15] M. X. Song, K. Chen, X. Zhang, and J. Wang, "The lazy greedy algorithm for power optimization of wind turbine positioning on complex terrain," *Energy*, vol. 80, pp. 567–574, Feb. 2015.

[16] J. Feng and W. Z. Shen, "Design optimization of offshore wind farms with multiple types of wind turbines," *Appl. Energy*, vol. 205, pp. 1283–1297, Nov. 2017.

[17] S. Y. D. Sorkhabi, D. A. Romero, G. K. Yan, M. D. Gu, J. Moran, M. Morgenroth, and C. H. Amon, "The impact of land use constraints in multi-objective energy-noise wind farm layout optimization," *Renew. Energy*, vol. 85, pp. 359–370, Jan. 2016.

[18] W. Y. Kwong, P. Y. Zhang, D. Romero, J. Moran, M. Morgenroth, and C. Amon, "Multi-objective wind farm layout optimization considering energy generation and noise propagation with NSGA-II," *J. Mech. Design*, vol. 136, no. 9, Sep. 2014, Art. no. 091010.

[19] M. A. Lackner and C. N. Elkinton, "An analytical framework for offshore wind farm layout optimization," *Wind Eng.*, vol. 31, no. 1, pp. 17–31, Jan. 2007.

[20] R. Shakoor, M. Y. Hassan, A. Raheem, and Y.-K. Wu, "Wake effect modeling: A review of wind farm layout optimization using Jensenś model," *Renew. Sustain. Energy Rev.*, vol. 58, pp. 1048–1059, May 2016.

[21] W.-Y. Ku, "Hybrid exact methods for solving strictly convex integer quadratic programs," Ph.D. dissertation, Dept. Mech. Ind. Eng., Univ. Toronto, 2017.

[22] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, "The ising model: Teaching an old problem new tricks," D-Wave Syst., Tech. Rep., 2010. Accessed: Jul. 22, 2022. [Online]. Available: https://www.dwavesys.com/media/vbklsvbh/weightedmaxsat_v2.pdf

[23] L. Davis, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[24] I. Ulku and C. Alabas-Uslu, "A new mathematical programming approach to wind farm layout problem under multiple wake effects," *Renew. Energy*, vol. 136, pp. 1190–1201, Jun. 2019.

[25] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of Constraint Programming*. Amsterdam, The Netherlands: Elsevier, 2006.

[26] J. S. González, A. G. G. Rodriguez, J. C. Mora, J. R. Santos, and M. B. Payan, "Optimization of wind farm turbines layout using an evolutive algorithm," *Renew. Energy*, vol. 35, no. 8, pp. 1671–1681, Aug. 2010.

[27] S. A. Grady, M. Y. Hussaini, and M. M. Abdullah, "Placement of wind turbines using genetic algorithms," *Renew. Energy*, vol. 30, no. 2, pp. 259–270, Feb. 2005.

[28] U. A. Ozturk and B. A. Norman, "Heuristic methods for wind energy conversion system positioning," *Electr. Power Syst. Res.*, vol. 70, no. 3, pp. 179–185, Aug. 2004.

[29] R. A. Rivas, J. Clausen, K. S. Hansen, and L. E. Jensen, "Solving the turbine positioning problem for large offshore wind farms by simulated annealing," *Wind Eng.*, vol. 33, no. 3, pp. 287–297, May 2009.

[30] M. Samorani, "The wind farm layout optimization problem," in *Handbook of Wind Power Systems*. Berlin, Germany: Springer, 2013, pp. 21–38.

[31] K. Yang and K. Cho, "Simulated annealing algorithm for wind farm layout optimization: A benchmark study," *Energies*, vol. 12, no. 23, p. 4403, 2019.

[32] A. Billionnet, "Different formulations for solving the heaviest K-subgraph problem," *INFOR, Inf. Syst. Oper. Res.*, vol. 43, no. 3, pp. 171–186, 2005.

[33] P. van Emde Boas, "Another NP-complete problem and the complexity of computing short vectors in a lattice," Dept. Math., Univ. Amsterdam, Tech. Rep., 1981.

[34] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958–2008*. Berlin, Germany: Springer, 2010, pp. 105–132.

[35] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, and J. Lee, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optim.*, vol. 5, no. 2, pp. 186–204, 2008.

[36] M. Tawarmalani and N. V. Sahinidis, "A polyhedral branch-and-cut approach to global optimization," *Math. Optim.*, vol. 103, no. 2, pp. 225–249, 2005.

[37] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Oper. Res.*, vol. 14, no. 4, pp. 699–719, Jul./Aug. 1966.

[38] J. E. Kelley, Jr., "The cutting-plane method for solving convex programs," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 4, pp. 703–712, 1960.

[39] R. Bixby and E. Rothberg, "Progress in computational mixed integer programming—A look back from the other side of the tipping point," *Ann. Oper. Res.*, vol. 149, no. 1, pp. 37–41, Feb. 2007.

[40] F. Furini, E. Traversi, P. Belotti, A. Frangioni, A. Gleixner, N. Gould, L. Liberti, A. Lodi, R. Misener, H. Mittelmann, N. V. Sahinidis, S. Vigerske, and A. Wiegele, "QPLIB: A library of quadratic programming instances," *Math. Program. Comput.*, vol. 11, no. 2, pp. 237–265, Jun. 2019, doi: 10.1007/S12532-018-0147-4.

[41] C. Coffrin, H. Nagarajan, and R. Bent, "Evaluating Ising processing units with integer programming," in *Proc. CPAIOR*, 2019, pp. 163–181.

[42] M. W. Johnson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, p. 194, 2011.

[43] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 667–672.

[44] Fujitsu Limited. (2021). *The Third Generation of the Digital Annealer*. Accessed: Aug. 20, 2021. [Online]. Available: https://www.fujitsu.com/jp/group/labs/en/documents/about/resources/tech/techintro/3rd-g-da_en.pdf

[45] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[46] A. Sheikholeslami, "The power of parallelism in stochastic search for global optimum: Keynote paper," in *Proc. IEEE 47th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2021, pp. 36–42.

[47] K. Hukushima and K. Nemoto, "Exchange Monte Carlo method and application to spin glass simulations," *J. Phys. Soc. Jpn.*, vol. 65, no. 6, pp. 1604–1608, Jun. 1996.

[48] E. Cohen, A. Senderovich, and J. C. Beck, "An Ising framework for constrained clustering on special purpose hardware," in *Proc. Int. Conf. Integr. Constraint Program., Artif. Intell., Oper. Res.*, 2020, pp. 130–147.

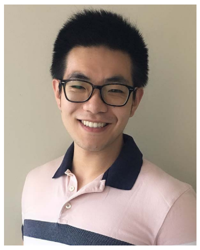
[49] H. Salehinejad and S. Valaee, "Ising-dropout: A regularization method for training and compression of deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3602–3606.

[50] Z. Naghsh, M. Javad-Kalbasi, and S. Valaee, "Digitally annealed solution for the maximum clique problem with critical application in cellular V2X," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[51] M. T. Rahman, S. Han, N. Tadayon, and S. Valaee, "Ising model formulation of outlier rejection, with application in WiFi based positioning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 4405–4409.

ARIK SENDEROVICH received the B.Sc. degree in industrial engineering and management, the M.Sc. degree in statistics, and the Ph.D. degree in data science from Technion, in 2006, 2012, and 2017, respectively.

He is an Assistant Professor with the School of Information Technologies (ITEC), York University. In 2017, he received the Lyon Sachs Scholarship (awarded to one Ph.D. grad per year) and worked as a Postdoctoral Fellow with the Toronto Intelligent Decision Engineering Laboratory (TIDEL), University of Toronto. He is also a Research Fellow with the TD-MDAL (Rotman School of Management), where he supervises student projects and organizes professional events. Prior to joining ITEC, he held the Assistant Professor positions with the Rotman School of Management and the Faculty of Information (both with the University of Toronto).



JIACHEN ZHANG received the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, in 2019, where he is pursuing currently the Ph.D. degree with the Department of Mechanical and Industrial Engineering. His research interests include specialized hardware, quadratic optimization, scheduling, and hybrid optimization methods with applications in artificial intelligence and operations research. His project can be found at https://www.overleaf.com/project/6202bcd5e5763669d14dc627

ELDAN COHEN received the Ph.D. degree from the Department of Mechanical and Industrial Engineering, University of Toronto, in 2020. After the Ph.D. degree, he worked as a Postdoctoral Fellow with the Department of Computer Science, University of Toronto; and the Vector Institute. He is an Assistant Professor with the Department of Mechanical and Industrial Engineering, University of Toronto. His research interests include machine learning; heuristic search and optimization; and scalable data mining with applications in automated planning, natural language processing, and software engineering.

J. CHRISTOPHER BECK is a Professor of industrial engineering with the Department of Mechanical and Industrial Engineering, University of Toronto; and the Director of the Toronto Intelligent Decision Engineering Laboratory. Since 2004, he has been supervising or co-supervising over 40 Ph.D. and master's students and published more than 140 papers in international venues. His research interests include constraint programming, mixed integer programming, AI planning and heuristic search, and scheduling. He is/will be the President of the Association for Constraint Programming, from 2021 to 2022; and served as the President of the Executive Committee for the International Conference on Automated Planning and Scheduling, from 2014 to 2016.