Efficient Gradient-Based Optimization for Joint Layout Design and Control of Wind Turbines

James Kotary
Pacific Northwest National Laboratory
Richland, WA
james.kotary@pnnl.gov

Natalie Isenberg
Pacific Northwest National Laboratory
Richland, WA
natalie.isenberg@pnnl.gov

Draguna Vrabie
Pacific Northwest National Laboratory
Richland, WA
draguna.vrabie@pnnl.gov

***Abstract*—A central challenge in the design of energy-efficient wind farms is the presence of wake effects between turbines. When a wind turbine harvests energy from free wind, it produces a turbulent region with reduced energy for downstream turbines. Strategies for increasing the efficiency of wind farms by mitigating wake effects have been the subject of much computational research. To this end, both the positioning of the turbines and their cooperative control must be taken into account. Since they are mutually dependent, increasing attention has been paid to the joint consideration of layout design and control. However, joint modeling approaches lead to large-scale and difficult optimization problems, which lack efficient solution strategies. This paper describes an efficient optimization framework for the joint design and steady-state control of wind turbines under a continuous wake effect model. Through a multi-level reformulation of the joint optimization problem, it shows how problem sub-structure can be efficiently exploited to yield order-of-magnitude reduction in convergence times over naive approaches and prior proposals.**

I. Introduction

Wind is considered one of the primary sources of renewable energy, along with solar and hydropower. In the year 2024, over 8.1 percent of global electricity was supplied by wind, and onshore wind in particular has become one of the least expensive energy sources available [1]. On the other hand, the energy productivity of a wind farm is highly dependent on its efficient design and control. A primary source of energy loss in wind farms comes due to aerodynamic interactions between turbines known as wake effects - when free wind energy is collected by a turbine, a turbulent region forms, leaving less energy available to further downstream turbines in its wake.

A significant body of research is dedicated to optimization models and methods aimed at minimizing the impact of wake effects on power production, via either its control strategies [2] or its layout [3]. For example, rotational control around the yaw axis allows for the redirection of wakes away from downstream turbines. Similarly, an optimal layout can position turbines to avoid the worst of their upstream counterparts' wake effects. Layout optimization has widely been considered in isolation from control considerations, but such an approach inherently makes simplifying assumptions about the subsequent control policy. Recent work [4] shows that increased power efficiency can be obtained by optimizing the wind farm's layout *jointly* with its steady-state control. However the joint layout and steady-state control optimization problems pose significant computational hurdles - they inherit the nonconvexity of the underlying layout problem, and rely on duplication of variables which leads to problem sizes that challenge conventional solvers. Furthermore, long convergence times of existing solvers hinder repeated solution strategies that could help overcome local minima.

To address those challenges, this paper presents a gradient-based optimization methodology for efficiently solving the joint wind farm layout and control problem. A specialized algorithm is proposed, based on repeated solution of efficiently solvable control subproblems, whose relation to a master layout problem is efficiently captured by well-defined gradients. Its efficiency advantage is demonstrated on an array of challenging test cases, which show orders-of-magnitude faster convergence over naive approaches on large problems. An open source optimization library comes with this paper, available at: https://github.com/anon-research-coding/WFLCopt.

II. Wind Farm Power Generation Model

This section describes the power production of a wind farm as a function of layout and control variables. We adopt a steady-state parametric wake effect model based on conservation of momentum which is widely used in layout optimization [4]–[6]. Let there be N_T turbines each of diameter D, to be placed within a feasible farm region $\mathcal{F} \subset \mathbb{R}^2$. The *layout* of the wind farm is defined by positional variables $\boldsymbol{l} \in \mathbf{R}^{N_T \times 2}$, within their joint feasible set $\mathcal{L} \coloneqq \mathcal{F}^{N_T}$. Turbines are primarily *controlled* via their yaw angles $\boldsymbol{\lambda} \in \mathbb{R}^{N_T}$ within feasible limits $\boldsymbol{\lambda} \in \Lambda$, which take the form of upper and lower bounds $\Lambda = \{\boldsymbol{\lambda} \text{ s.t. } \lambda_{min} \leq \boldsymbol{\lambda} \leq \lambda_{max}\}$ on the rotational angle of each turbine. The layout of a wind farm is considered to be fixed once decided, while the controls allow for adaptation in response to changing wind conditions. We follow the typical modeling approach where wind conditions and associated control states are modeled in the steady state, meaning that dynamic control between states is not explicitly considered.

The *power* generated by the wind farm under *freestream wind direction* θ^ω, given positions $\boldsymbol{l}$ and yaws $\boldsymbol{\lambda}$, is denoted $P(\boldsymbol{l}, \boldsymbol{\lambda}, \theta^\omega)$. Its value depends on the effective wind speed at each turbine location, which must account for wake effects

Supported by the Energy Earthshots Initiative as part of the DOE Office of Biological and Environmental Research. PNNL is operated by DOE by the Battelle Memorial Institute under Contract DE-A06-76RLO 1830.

979-8-3315-8138-1

from all other turbines. For notational convenience we specify two vectors of independent coordinates $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^{N_T}$ so that $\boldsymbol{l} = [\boldsymbol{x}; \boldsymbol{y}] \subset \mathbf{R}^{N_T \times 2}$ is made up of their concatenation. We define additional coordinate systems whose canonical directions are aligned with, and perpendicular to a fixed wind angle θ^ω. For any two turbines indexed i and j, we denote their relative angles in Cartesian space as θ_{ij} and define their inter-distances along the *downstream* and *radial* directions respectively:

$$d_{ij} = \|l_i - l_j\|_2 \cos(\theta_{ij} - \theta^\omega), \quad r_{ij} = \|l_i - l_j\|_2 \sin(\theta_{ij} - \theta^\omega),$$

where dependence of d_{ij} and r_{ij} on the wind angle index ω is omitted from their notation, being apparent from context throughout. Corresponding matrix variables $\boldsymbol{d}, \boldsymbol{r} \in \mathbb{R}^{N_T \times N_T}$ are made up of those scalar values. They represent relative coordinates, induced by the wind direction and centered at each turbine, and serve simplify the modeling of wake effects.

A. Wake Effect Model

Wake effects are measured in terms of their resulting windspeed deficits, relative to the *freestream wind speed* U. At each point in $2D$ space, wake effects from upstream turbines propagate primarily along the downstream direction $\boldsymbol{d}$, with their deficit magnitudes decaying along the radial direction $\boldsymbol{r}$ in a Gaussian profile. Let $WD(\boldsymbol{d}, \boldsymbol{r})$ represent the pairwise *windspeed deficit factors* due to wake effects between turbines:

$$WD(\boldsymbol{d}, \boldsymbol{r}) = \left(1 - \sqrt{1 - \frac{\sigma_{\boldsymbol{r}0}}{\sigma_{\boldsymbol{r}}} \boldsymbol{C_T}}\right) \exp\left(\frac{(\boldsymbol{r} - \boldsymbol{\delta}_f)^2}{2\boldsymbol{\sigma}_{\boldsymbol{r}}^2}\right), \quad (1)$$

where $\boldsymbol{C_T}$ are thrust coefficients. These define Gaussian wake profiles which expand linearly in the downstream direction at rate $k_{\boldsymbol{r}}$. That is, $\boldsymbol{\sigma}_{\boldsymbol{r}} = \boldsymbol{\sigma}_{\boldsymbol{r}0} + k_{\boldsymbol{r}}\boldsymbol{d}$ where $\boldsymbol{\sigma}_{\boldsymbol{r}0} = \frac{D}{2\sqrt{2}} \cos(\boldsymbol{\lambda})$.

Importantly, wind turbines can only induce wake effects on downstream positions. Correspondingly, formula (1) holds only for elements where $d_{ij} > 0$; otherwise, the deficit factor is typically defined to be zero [4], [5]. This leads to potential discontinuities in the function (1) which are physically unrealistic and numerically challenging. These can be closely approximated everywhere by a smooth function $\widetilde{WD}(\boldsymbol{d}, \boldsymbol{r}) \coloneqq s(\boldsymbol{d}) \cdot WD(\boldsymbol{d}, \boldsymbol{r})$, where $s(\boldsymbol{d}) = \frac{1}{1+e^{-\tau \boldsymbol{d}}}$ is a sigmoid function, noting that $WD = \widetilde{WD}$ as $\tau \to \infty$. In addition to its linear expansion, the wake profile also experiences deflection of its centerline by the value $\boldsymbol{\delta}_f$, also a function of $\boldsymbol{d}$:

$$\boldsymbol{\delta}_f = a_d D + b_d \boldsymbol{d} + \frac{\phi}{5.2} E_0 \sqrt{\frac{\sigma_{\boldsymbol{r}0}}{k_{\boldsymbol{r}} C_T}} \ln\left[\frac{(1.6 + \sqrt{C_T})(1.6\sqrt{\sigma_r/\sigma_{\boldsymbol{r}0}} - \sqrt{C_T})}{(1.6 - \sqrt{C_T})(1.6\sqrt{\sigma_r/\sigma_{\boldsymbol{r}0}} + \sqrt{C_T})}\right] \quad (2a)$$

where we define intermediate variables: $\boldsymbol{E}_0 = \boldsymbol{C}_0^2 - 3e^{1/12}\boldsymbol{C}_0 + 3e^{1/3}$ and $\boldsymbol{C}_0 = 1 - \sqrt{1 - \boldsymbol{C}_T}$, plus $\phi = \frac{0.3\boldsymbol{\lambda}}{\cos \boldsymbol{\lambda}}(1 - \sqrt{1 - \boldsymbol{C}_T \cos \boldsymbol{\lambda}})$ following [4]. Detailed formulations of the deflection model are found in [7], and a_d, b_d and $k_{\boldsymbol{r}}$ are physical parameters.

B. Effective Wind Speed and Power Production

The amount of power that can be harvested by turbine i is determined by the *effective windspeed* V_i at its location, which results from the *combined* wake effects from all turbines in the wind farm. To combine multiple wake effects at a single point, we adopt the traditional approach which determines the total windspeed deficit as the Euclidean norm over those induced by the individual turbines [4]–[6]:

$$V_i = U \cdot \left(1 - \sqrt{\sum_{j \neq i} \widetilde{WD}(d_{ji}, r_{ji})^2}\right). \quad (3)$$

The total power generated by the wind farm is the sum over that of each turbine, due to its effective wind speed:

$$P(\boldsymbol{l}, \boldsymbol{\lambda}, \theta^\omega) = \sum_{i=1}^{N} \frac{1}{2}\rho(\frac{\pi}{4}D^2) C_{P_i} \cos(\lambda_i) V_i^3. \quad (4)$$

The coefficients $\boldsymbol{C_P}$, along with $\boldsymbol{C_T}$ in (1), depend on controllable axial induction factors α_i for each turbine. While nothing prevents their optimization along with $\boldsymbol{\lambda}$, they are considered less effective than yaw control [8], [9] and often set to the constant value $\alpha_i = 1/3$, which we adopt.

Readers interested in further details and derivations of the power generation models are referred to [5], [7]. The goal of this work is optimization of power objectives based on (4), for which it suffices to recognize that (4) is a smooth but highly nonconvex function of layout and control variables $(\boldsymbol{l}, \boldsymbol{\lambda})$.

III. Joint Optimization Problem

The aim of this paper is to develop optimization methods for solving a *joint* layout and control problem, which can be viewed as an optimization problem in two stages. The first-stage decision is permanent and determines the layout $\boldsymbol{l} \in \mathcal{L}$, while the second determines control variables $\boldsymbol{\lambda} \in \Lambda$. Their roles are distinguished by the fact that $\boldsymbol{\lambda}$ can be adjusted in response to changing wind conditions θ^ω. To quantify those changing conditions, we assume that θ^ω are realized from an underlying continuous random variable Θ. Like prior works [4]–[6], we approximate Θ by deriving from it a discrete random variable in order to form a tractable optimization problem. First, the range $[0^o, 360^o]$ is partitioned into W equal-sized bins. Each ω^{th} bin is represented by its midpoint, denoted θ^ω for $1 \leq \omega \leq W$. These wind directions define W distinct *scenarios*, each in which *scenario-wise turbine control variables* $\boldsymbol{\lambda}^\omega$ may be optimized in response. Wind directions θ^ω are sampled from Θ, and the frequency of each bin is used to construct the associated scenario probabilities p^ω. All wind directions are relative to a fixed reference angle at due East.

The overall *joint layout and control problem* seeks a single assigment of turbine positions $\boldsymbol{l}$, plus yaw angles $\boldsymbol{\lambda}^\omega$ for each of W wind directions θ^ω, which maximize the expected power:

$$\max_{\boldsymbol{l}, \boldsymbol{\lambda}^\omega \, \forall \omega} \sum_{\omega=1}^{W} p^\omega P(\boldsymbol{l}, \boldsymbol{\lambda}^\omega, \theta^\omega) \quad (5a)$$

$$s.t. \quad \boldsymbol{l} \in \mathcal{L} \quad (5b)$$

$$\boldsymbol{\lambda}^\omega \in \Lambda \qquad \forall \omega \quad s.t. \;\; 1 \leq \omega \leq W \quad (5c)$$

$$\|\boldsymbol{l}_i - \boldsymbol{l}_j\|_2^2 \geq 4D \quad \forall i, j \;\; s.t. \;\; 1 \leq i < j \leq N_T. \quad (5d)$$

In addition to the wind farm perimeter (5b) and control bounds (5c), inter-distance constraints (5d) ensure that wind turbines are spaced at a minimum safe distance. We summarize the

main challenges posed by problem (5) as follows: **(1)** Its variable count scales quadratically as $N_T \cdot (W + 2)$, quickly leading to problem sizes which challenge conventional solvers and **(2)** It is nonconvex in both objective and constraints, with flat regions and sharp inclines prevalent in each term of (5a), while nonconvex inter-distance constraints number quadratically in N_T. On the other hand, the majority of variables in (5) are scenario-wise variables $\boldsymbol{\lambda}^\omega$, which are mutually uncoupled and subject only to simple bounds. The following proposal leverages the special structure in Problem (5) to arrive at a framework for its efficient optimization.

IV. Multilevel Reformulation and Solution

In this section, we present a computational framework for building efficient optimization methods to solve problem (5). We first observe that the optimal control variables $\boldsymbol{\lambda}^\omega$ in each scenario are directly determined by the layout $\boldsymbol{l}$, and the corresponding power production can be differentiated as a function of $\boldsymbol{l}$. The *optimal power function* $P^\star$ below represents the available power as a function of position, given that yaws are optimized with respect to the wind direction θ^ω:

$$P^\star(\boldsymbol{l}, \theta^\omega) \coloneqq \max_{\boldsymbol{\lambda} \in \Lambda} \quad P(\boldsymbol{l}, \boldsymbol{\lambda}, \theta^\omega). \tag{6}$$

The joint problem (5) is then equivalent to the following:

$$\max_{\boldsymbol{l} \in \mathbb{R}^{2 \cdot N_T}} \quad \sum_{\omega=1}^{W} p^\omega P^\star(\boldsymbol{l}, \theta^\omega) \tag{7a}$$

$$s.t. \quad \boldsymbol{l} \in \mathcal{L} \tag{7b}$$

$$\|\boldsymbol{l}_i - \boldsymbol{l}_j\|_2^2 \geq 4D \quad \forall i, j \; s.t. \; 1 \leq i < j \leq N_T. \tag{7c}$$

Problems (6),(7) can be viewed as the *inner* and *outer problems* in a *multilevel reformulation* of problem (5). While equivalent to the original joint problem (5), it poses different challenges. Each of its objective terms $P^\star(\boldsymbol{l}, \theta^\omega)$ requires (nonconvex) optimization to evaluate, making the overall objective function expensive to evaluate. The lack of a closed form for each $P^\star(\boldsymbol{l}, \theta^\omega)$ also prevents direct application of gradient-based optimization solvers to (7).

On the other hand, this formulation is expressed solely in terms of the $2N_T$ positional variables $\boldsymbol{l}$, which number a factor of W less than those of the joint formulation. The present proposal is based on directly solving the smaller-scale problem (7) in place of the original joint problem (5), while at the same time mitigating the computational challenges of its objective function. It relies on two main concepts: **(1)** exact differentiation of $P^\star$ via envelope theorems and **(2)** an efficient solution scheme for repeated resolution of the inner problems (6). A computational pipeline composed of these techniques enables fast, repeated evaluation of the composite objective (7a) and its gradients, which can be paired with various appropriate nonconvex solvers to directly optimize problem (7). Each are detailed in the following subsections.

A. Differentiation of the Implicit Objective Function

Let $f(\boldsymbol{l}) \coloneqq \sum_{\omega=1}^{W} p^\omega \cdot P^\star(\boldsymbol{l}, \theta^\omega)$ denote the objective function in problem (7). Its gradient is found by combining the gradients of W implicitly defined optimal power functions:

$$\nabla_{\boldsymbol{l}} f(\boldsymbol{l}) = \sum_{\omega=1}^{W} p^\omega \cdot \nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega) \tag{8}$$

An efficient formula for each constituent term $\nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega)$ is provided by the Envelope Theorems for differentiation of a continuous minimizer [10], which reduce its derivative to a standard partial derivative of the underlying power function P, evaluated at its corresponding argmin. Define $\boldsymbol{\lambda}^\star = \operatorname{argmax}_{\boldsymbol{\lambda} \in \Lambda} P(\boldsymbol{l}, \boldsymbol{\lambda}, \theta^\omega)$ so that $\boldsymbol{\lambda}^\star = \operatorname{argmax}_{\boldsymbol{\lambda} \in \Lambda} P(\boldsymbol{l}, \boldsymbol{\lambda}, \theta^\omega)$, then

$$\nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega) = \nabla_{\boldsymbol{l}} P(\boldsymbol{l}, \boldsymbol{\lambda}^\star, \theta^\omega). \tag{9}$$

That is, to compute the gradient $\nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega)$, one first solves the inner problem (6) for $P^\star(\boldsymbol{l}, \theta^\omega)$. The associated argmin $\boldsymbol{\lambda}^\star$ comes as a byproduct, at *no additional cost*. The remaining standard gradient $\nabla_{\boldsymbol{l}} P(\boldsymbol{l}, \boldsymbol{\lambda}^\star, \theta^\omega)$ in (9) is easily computed, e.g., by automatic differentiation (see details in Section V).

Note that $\nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega)$ could be approximated by finite differences [11], which is the default behavior of some black-box optimization solvers. However, this requires solving problem (6) once for each of its N_T components. Thus, approximation of $\nabla_{\boldsymbol{l}} f(\boldsymbol{l})$ requires N_T^2 solver calls. By comparison, this result provides exact gradients with no additional solver calls.

Theoretical Remarks: We remark that the validity of (9) requires P to be continuous and have a unique partial minimizer $\lambda^\star$ [10]. Continuity of P follows from its construction as a composition of continuous functions [5], but nonunique $\lambda^\star$ may arise since total power is unaffected by the yaw angles of the most-downstream turbines, since they induce no wake effects. This issue is resolved by adding a tie-breaking term to the power objective, which has negligible effects on the solution but ensures uniqueness. While nonconvexity of P prevents guarantees that $\nabla_{\boldsymbol{l}} P^\star(\boldsymbol{l}, \theta^\omega)$ exists everywhere, it is typical in nonconvex settings to prove that it exists almost-everywhere, and that formula (9) is valid whenever it exists [10]. A detailed analysis on the existence of the gradient (9) is reserved for an extension of this work.

B. Fast Solution and Updating Scheme for the Inner Problems

A generic optimization algorithm repeatedly updates an estimate of the decision variable $\boldsymbol{l}$, as a deterministic function U of the current estimate, plus its corresponding objective value $f(\boldsymbol{l})$ and gradient $\nabla_{\boldsymbol{l}} f(\boldsymbol{l})$. For this we define notation:

$$\boldsymbol{l} \coloneqq U\left(\, \boldsymbol{l}, f(\boldsymbol{l}), \nabla_{\boldsymbol{l}} f(\boldsymbol{l}) \,\right). \tag{10}$$

Recall that the multilevel reformulation (7) gains a drastic reduction in problem size from the joint problem (5), but at the cost of complicating the evaluation of the arguments $f(\boldsymbol{l})$ and $\nabla_{\boldsymbol{l}} f(\boldsymbol{l})$ to the update routine (10). To alleviate that tradeoff, this section thus presents a computational pipeline that provides fast, repeated evaluations of the optimal power functions $P^\star(\boldsymbol{l}, \theta^\omega)$ and thus also the total objective $f(\boldsymbol{l})$ in problem (7). In principle, the framework is solver-agnostic and can be paired with any optimization method represented by (10) which accomodates nonconvex objectives and constraints.

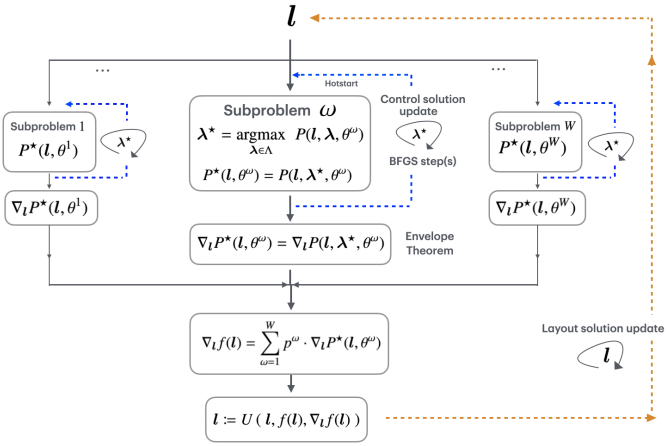

Fig. 1: Illustrating the combined computational framework.

A key feature of the reformulation (7) is that its inner subproblems (6) require nonconvex optimization, but subject only to simple bounds, a.k.a. box constraints $\Lambda = \{\boldsymbol{\lambda} : \lambda_{\min} \leq \boldsymbol{\lambda} \leq \lambda_{\max}\}$. Meanwhile the outer problem subject to arbitrary constraints, leading to a form reminiscent of the dual reformulations employed by practical augmented Lagrangian solvers such as LANCELOT [12]. Such methods exploit efficient handling of the inner problem by a limited-memory box-constrained Broyden–Fletcher–Goldfarb–Shanno (L-BFGS-B) algorithm [13], a choice driven by two major advantages: **(1)** It is proven to excel in optimizing nonconvex functions over box constraints and **(2)** it makes highly efficient use of hot-starts when repeated solutions are desired [12].

Our framework applies a similar strategy, but to each of the constituent subproblems (6) within the multilevel reformulation (7). Figure 1 illustrates an overall schematic based on the equations above. Each iteration of the outer step (10) updates location variables $\boldsymbol{l}$ in the outer problem (7), shown in orange. Within each outer step, re-evaluation of $f(\boldsymbol{l})$ requires re-solving each of W inner problems (6) by iterating L-BFGS-B to convergence, shown in blue. This yields $P^\star(\boldsymbol{l}, \theta^\omega)$ for $1 \leq \omega \leq W$. Each re-solve is accelerated by hotstarting the BFGS algorithm from a cached copy of its optimal solution $(\boldsymbol{\lambda}^\star)^\omega$ from the previous outer iteration. Since the outer iterations (10) result in only small changes to $\boldsymbol{l}$, successive evaluations of $f(\boldsymbol{l})$ converge in very few iterations, reducing their computation time by over 99% in our experiments. Following Section IV-A, $\nabla_{\boldsymbol{l}} f(\boldsymbol{l})$ is then efficiently obtained by computing the standard derivatives (9) with respect to each subproblems and combining them per equation (8). The next section shows empirically how it enables order-of-magnitude reductions in solving time when paired with off-the-shelf optimization solvers to implement the iterations (10).

V. Implementation and Experiments

In this section, we evaluate the ability of our proposed computational framework to improve upon the efficiency of conventional optimization solvers applied to the joint layout and control problem (5). Section V-A discusses implementation details regarding each method evaluated. An array of increasingly difficult test problems is described in Section V-B, and a systematic comparison of results is presented in VI.

A. Methods and Implementation

The computational framework described in Section IV is independent of any particular optimization algorithm used to solve the outer problem (7), which so far has been generically represented by equation (10). When considering the choice of algorithm, it must be capable of handling general nonlinear programs with nonconvex constraints and objective. It should also be equipped with robust stepsize control, including advanced heuristics such as trust region and line search methods. We consider two main classes of algorithms which share these properties: **(1)** the *interior point methods* (IP), where U in (10) solves a log-barrier approximation to the true problem at each step and **(2)** *sequential quadratic programming (SQP)*, where U solves a quadratic approximation to the problem. We refer to these as the *base methods*. We apply them to each *problem form*: **(1)** the original joint form (5) (called JOINT), and **(2)** the multilevel reformulation (7) augmented by the techniques proposed in Section (IV) (called MLR-A). Each problem form / base method combination is evaluated, totaling four. The JOINT variants *serve as baselines* in this study, having been previously proposed for isolated layout optimization [5], [6] and tested on the joint layout and control problem [4].

Implementation: All optimization methods tested in this section are implemented using the SciPy suite of solvers. Its interior point method is a trust-region IP variant known as NITRO [14]. Its main SQP method is a variant called SLSQP [15]. Finally, our MLR-A methods are built using SciPy's implementation of L-BFGS-B, based on the proposal of [13]. All solvers are given an objective tolerance of $1e-5$, which creates a fair standard for comparing convergence times between equivalent objectives (5a) and (7a). All other settings use SciPy's standard values. Derivatives are calculated by automatic differentiation in PyTorch, including (9).

On Further Baselines: We remark that prior work [4] proposes a decomposition scheme for solving (5), but reports no speedups over naive SQP on the JOINT formulation. The included library contains our Python implementation of [4]. Due to this lack of runtime advantage, plus inconsistent convergence in our studies, a systematic comparison with [4] is reserved for an extension of this work.

B. Problem Specifications

To evaluate the scalability of each method, we solve a sequence of joint layout and control problems in which $N_T \in \{25, 36, 49, 64, 81, 100\}$ increases by perfect squares. In each case, $W = 36$ is held constant, and θ^ω are sampled for $1 \leq \omega \leq W$ from a Dirichlet distribution with $\alpha = 1$. Feasible turbine positions are in $\mathcal{F} = [0, x_{max}] \times [0, y_{max}]$. We set $x_{max} = y_{max} := 1.3\sqrt{N_T} \cdot 4D$ to standardize the turbine density across cases. Problems (5) and (7) are prone to local optima, which have a randomizing effect on both the convergence time and final objective value. Thus, all Runtime and Objective

TABLE I: Algorithm Runtimes and Objective Values Across Problem Sizes

Form	Method	Number of Turbines N_T											
		Runtime (s)						Objective Value (GWh)					
		25	**36**	**49**	**64**	**81**	**100**	**25**	**36**	**49**	**64**	**81**	**100**
JOINT	IP	3719	10147	29288	57565	✗	✗	475.1	672.9	899.9	1157.5	✗	✗
	SQP	1225	4315	15363	52469	109790	✗	**476.9**	674.1	903.3	1163.0	1457.2	✗
MLR-A	IP	3404	4780	7414	14795	27837	55954	475.6	673.5	902.2	1162.2	1454.2	1773.6
	SQP	**364**	**569**	**933**	**1873**	**3353**	**6976**	**476.9**	**674.8**	**904.5**	**1165.6**	**1457.9**	**1780.3**

TABLE II: Physical Parameters

D	ρ	U	a_d	b_d	k_r	$\lambda_{\min}$	$\lambda_{\max}$
126 m	1.23	8	-0.035	-0.01	0.03	$-30°$	$30°$

values are averaged over 30 independent, randomly initialized solver runs.

VI. Results and Conclusions

Table I reports the Runtime of each solution in seconds, along with the Objective Value in GWh, after multiplying the nominal power objective (5a),(7a) by $T = 8760^{hr}/_{yr}$ to yield annual energy. The best results across each combination of problem form and base method are marked in bold. Runtimes are further illustrated in Figure 2 as a function of problem size. A *time limit* of 36hr is imposed on each run. Runs that exceed that limit are marked by ✗ and omitted from the Figure.

Figure 2 shows how both the SQP (at left) and IP (at right) variants of MLR-A gain an increasing runtime advantage over their baseline JOINT counterparts. Both IP and SQP exceed the timeout threshold on the JOINT problem, past $N_T = 64$ and $N_T = 81$ respectively. Meanwhile the MLR-A approach converges on all problem sizes when paired with both the IP and SQP base methods. Regarding objective value, the MLR-A approach also consistently reports a marginal advantage, giving best results when paired with SQP. We speculate that the reduction of variable space leads to fewer local optima, but this effect is secondary to the runtime advantage.

We conclude that MLR-A should be combined with SQP for maximum efficiency. The comparison with IP methods serves as an ablation which supports this finding, which is consistent with previous findings which favor SQP for layout optimization. Our MLR-A variant (dark blue) converges up to 33 times faster than the standard SQP approach (light blue) while also gaining in the objective value.

Concluding Remarks: This study shows the potential for specialized optimization strategies to enhance automated design of wind farm layouts. The resulting runtime reductions can now enable various repeated solution strategies to overcome local minima, and may open the door to integration with metaheuristic frameworks. Generalization to floating offshore and other settings are directions for future work.

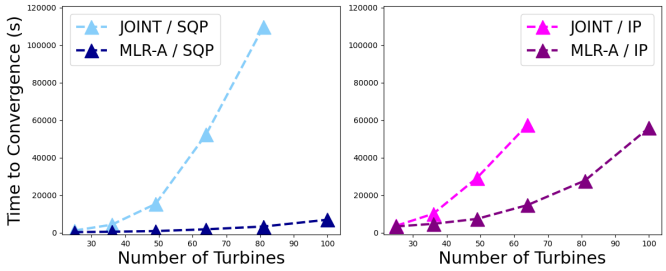

Fig. 2: Runtime vs Problem Size: SQP (left) and IP (right) variants, applied directly to problem (5) vs MLR framework.

References

[1] Y. Abdelilah, H. Bahar, T. Criswell, P. Bojek, F. Briens, and P. Feuvre, "Renewables 2020: Analysis and forecast to 2025," *IEA*, 2020.

[2] S. Boersma, B. M. Doekemeijer, P. M. Gebraad, P. A. Fleming, J. Annoni, A. K. Scholbrock, J. A. Frederik, and J.-W. van Wingerden, "A tutorial on control-oriented modeling and control of wind farms," in *2017 American control conference (ACC)*. IEEE, 2017, pp. 1–18.

[3] S. Pranupa, A. Sriram, and S. Nagaraja Rao, "A review of wind farm layout optimization techniques for optimal placement of wind turbines," *International Journal of Renewable Energy Research*, 2023.

[4] K. Chen, J. Lin, Y. Qiu, F. Liu, and Y. Song, "Joint optimization of wind farm layout considering optimal control," *Renewable Energy*, vol. 182, pp. 787–796, 2022.

[5] J. Park and K. H. Law, "Layout optimization for maximizing wind farm power production using sequential convex programming," *Applied energy*, vol. 151, pp. 320–334, 2015.

[6] D. Guirguis, D. A. Romero, and C. H. Amon, "Toward efficient optimization of wind farm layouts: Utilizing exact gradient information," *Applied energy*, vol. 179, pp. 110–123, 2016.

[7] M. Bastankhah and F. Porté-Agel, "Experimental and theoretical study of wind turbine wakes in yawed conditions," *Journal of Fluid Mechanics*, vol. 806, pp. 506–541, 2016.

[8] D. van der Hoek, S. Kanev, J. Allin, D. Bieniek, and N. Mittelmeier, "Effects of axial induction control on wind farm energy production-a field test," *Renewable energy*, vol. 140, pp. 994–1003, 2019.

[9] P. Fleming, J. Annoni, J. J. Shah, L. Wang, S. Ananthan, Z. Zhang, K. Hutchings, P. Wang, W. Chen, and L. Chen, "Field test of wake steering at an offshore wind farm," *Wind Energy Science*, vol. 2, no. 1, pp. 229–239, 2017.

[10] P. Milgrom and I. Segal, "Envelope theorems for arbitrary choice sets," *Econometrica*, vol. 70, no. 2, pp. 583–601, 2002.

[11] G. Froese, S. Y. Ku, A. C. Kheirabadi, and R. Nagamune, "Optimal layout design of floating offshore wind farms," *Renewable Energy*, vol. 190, pp. 94–102, 2022.

[12] A. R. Conn, G. Gould, and P. L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer Science & Business Media, 2013, vol. 17.

[13] D. Liu and J. Nocedal, "On the limited memory method for large scale optimization: Mathematical programming b," 1989.

[14] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[15] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.