Multi-Objective Microgrid Design Optimization

May Beebe, Ethan Hansen, Will Hogan, Nils Johnson

Colorado School of Mines, EENG521 Numerical Optimization, Dr. Michael Wakin

May 7, 2025

I. Introduction

A. Motivation

As climate change accelerates and power consumption grows, there is a critical need for efficient electrical grid designs. Designing energy systems requires optimizing reliability, costs, and sustainability. Renewable sources, though essential for sustainability goals, require special consideration as renewables introduce uncertainty in power generation due to weather variability [1]–[6]. To address these challenges, this project employs numerical optimization to design a cost-effective, low-carbon renewable energy system, optimizing the sizes and types of energy generation and storage technologies within an islanded microgrid.

B. Problem Statement

Optimize the sizing and selection of power generation and storage components of an islanded microgrid that minimizes both costs and greenhouse gas emissions within realistic environmental, operational, and demand constraints, ensuring reliable and sustainable energy supply.

II. Optimization Problem Formulation

The problem is formulated as a multi-objective optimization problem over the decision vector $x = [x_s, x_w, x_d, x_b]^T$ where $x_{s,w,d,b}$ represents installed capacity of solar PV, wind turbines, diesel generators, and battery storage (in kW or kWh). Bounds, reflecting engineering feasibility and field-validated deployment constraints [1]–[4], were imposed on all decision variables

$$0 \leq x_i \leq 4000$$

The goal is to simultaneously minimize two conflicting objectives: total system cost and lifetime diesel emissions, subject to system constraints on reliability and feasibility. The cost objective function is given by

$$f_c(x) = r(C_{\text{cap}}(x) + C_{\text{om}}(x) + C_{\text{df}}(x)) + C_{\text{Lp}}(x)$$

where r is project lifetime (years) and $C_{\text{cap,om,df,Lp}}$ are costs for installation, operation and maintenance, diesel fuel, and unmet demand penalty. The emissions objective function is defined as

$$f_{em}(x) = E_d(x)\gamma_{em}$$

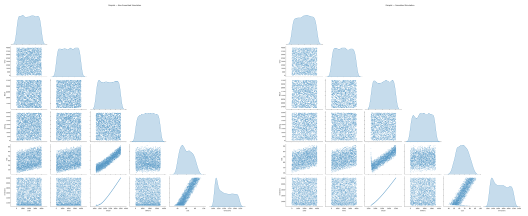
where $E_d =$ is the total diesel energy consumed and $\gamma_{em} =$is the diesel emissions factor in $\mathrm{kg\,CO_2/kWh}$. The combined objective function is therefore expressed as

$$\min_{x \in \mathbb{R}^4_+} \boldsymbol{f(x)} = [f_c(x),\ f_{em}(x)]^\top$$

To ensure reliability, we defined the constraint of a hard upper bound on the annual unmet load

$$U(x) = \sum_{t=1}^{8760} \max(0, D_t - P_t(x)) \leq 1 \text{ MWh}$$

where U_t denotes load demand at hour t, $U(x)$ is the maximum allowable unmet load per year, and $P_t(x)$ is the total power available at time t. Decision variables are defined via hourly power contributions from solar $[P_s(t)]$, wind $[P_w(t)]$, diesel$[P_d(t)]$, and battery discharge $[P_b^-(t)]$ are computed dynamically within a physics-based simulation that models hourly operation of the microgrid system over a one-year horizon. It includes component-level dynamics, such as battery charging constraints, diesel generator minimum load constraints, and intermittent renewable availability. Formal descriptions[1]were derived, however the internal operational rules and non-linear behaviors embedded in the simulation are enforced implicitly through conditional logic and dynamic state updates. These non-decision parameters are fixed and system-level specifications that define technology behavior, environmental conditions, and economic assumptions. Values used for these variables were either known industry standards or found via research [1], [7]–[10]. While these parameters are not being optimized, they define how the system behaves under different configurations, and therefore indirectly influence the objectives' outcomes and constraint feasibility.



(a) Non-smoothed (b) Smoothed

Fig. 1: Pairplot comparison of non-smoothed and smoothed simulation datasets.

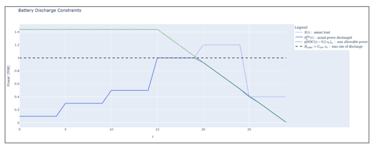


Fig. 2: Battery discharge governed by C-rate and SOC bounds.

Renewable energy sources, being governed by weather dynamics, add another layer of complexity to a problem landscape shaped by inter-temporal, non-linear constraints arising

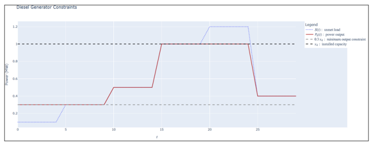


Fig. 3: Diesel generator enforces minimum output threshold.

from the system's operational component behaviors. Battery operation is limited by power and energy constraints, charging only up to its rate limit when surplus energy is available and discharging until constrained by rate or state-of-charge, as shown in **Figure 2**. Diesel generators add dispatch rigidity: once on, they must run above a minimum power and risk generating a surplus. **Figure 3** illustrates this step-wise behavior. A sweep of minimum threshold values (10–90%) in **Figure 4** and **Figure 5** shows that higher thresholds worsen cost and emissions, while lower thresholds offer more flexible backup, underscoring key system trade-offs. These simulation-driven insights illustrate the complex dynamics and trade-offs that emerge from component-level constraints and interactions within a hybrid microgrid.

To better understand the relationships among and between decision variables and objective function values, two datasets—one using hard logic and one with smooth approximations—were generated. **Figure 1** shows how capacity sizes, cost, and emissions correlate: Diesel capacity strongly drives both metrics due to its high fuel and carbon costs, while moderate battery sizing adds little cost, and wind and solar contributions are context-dependent, shaped by their timing relative to demand and storage. Importantly, this reveals that smoothing operational logic reduces discontinuities, permitting the use of gradient-based optimization methods.

III. Multi-Objective Optimization Strategies

A. *Formulation of a General Multi-Objective Problem*

The goal of multi-objective optimization is to minimize several objective functions simultaneously. The objective functions can be grouped into a single vector-valued objective function **f** where

$$\boldsymbol{f}(\boldsymbol{x}) = (f_1(x), f_2(x), ..., f_k(x))^T : \mathbb{R}^n \rightarrow \mathbb{R}^k$$

The minimization can be formulated as follows where $\mathbb{R}$ is the feasible set of x.

$$\min_{x \in R} \boldsymbol{f}(\boldsymbol{x})$$

Optimality points-efficient points-are points that produce the lowest value of all components of the objective function.

$$\boldsymbol{f}(\boldsymbol{x}^*) \leq \boldsymbol{f}(\boldsymbol{x})$$

There is an additional notion of comparison in multi-objective optimization problems with the concept of "domination". A point x^1 is said to dominate point x^2 if $x^1 \neq x^2$ and $\boldsymbol{f}(x^1) \leq \boldsymbol{f}(x^2)$ (i.e. point x^1 is better in every way than point x^2). In all multi-objective problems of interest, the different objective functions are competing in the sense that minimizing one objective function causes a rise in the other. Additionally, constraints on x further limit the values $\boldsymbol{f}$ can take. Often the value the optimizes all components of the objective function is infeasible or not possible. It is then of interest to look at the Pareto Optimal Front or Pareto Frontier. The Pareto Front is the set of non-dominated points and represents the design trade-off region where any performance increase in one objective function necessitates a degradation of performance in at least one part of the objective function [11] . By looking at the Pareto Front, a designer can analyze trade-offs and make decisions based on other information such as component availability or design timeline.

B. *Weighted Sum Algorithm*

The weighted sum, also known as linear scalarization, method is a straightforward approach to converting a multi-objective optimization problem into a single-objective optimization problem [11] . As the name implies, the solution involves weighting each function and summing the output of these weighted function. Mathematically this is expressed as:

$$\min_{x \in X} \sum_{i=1}^{k} w_i f_i(x)$$

where:

- x = decisions variables (solar capacity, wind capacity, diesel capacity, battery size)
- X = feasible set (constraints on x)
- $f_i(x)$ = individual objective functions (cost, emissions)
- w_i = weights assigned to each objective

In this microgrid optimization problem we formulate the optimization problem as:

$$\min_{x \in X} (w_c f_c(x) + w_{em} f_{em}(x))$$

In order to generate the Pareto front [2], the weights can be swept, and optimization performed for each weight combination:

$$0 < w_c < 1$$

$$w_{em} = 1 - w_c$$

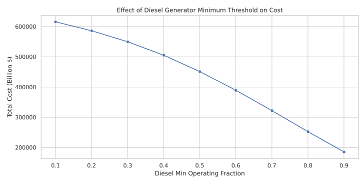


Fig. 4: Impact of diesel generator threshold on system cost.

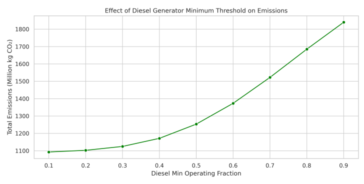


Fig. 5: Total emissions vs diesel generator minimum threshold.

While running the weighted sum algorithm, it was found to be sensitive to, and get stuck at local minima, so a multi-start approach was used to test random initializations for each weighting combination, and use the best result to form the algorithm's pareto front.

C. *Epsilon Constraint Algorithm*

The epsilon constraint method is another scalarization method that optimizes one objective function while holding all others as constraints.

$$\min_{x \in X} f_k(x) \quad \text{subject to} \quad f_{i \neq k}(x) \leq \epsilon$$

Each optimum value found by the constrained single variable optimization is a non-dominated point along the Pareto front [11], [12]. For the microgrid optimization problem, this method was executed with emissions as a constraint and a second time with cost as a constraint.

D. *Genetic Algorithm*

Genetic algorithms (GA) are a class of derivative free algorithms that perform well on multi-objective functions, and particularly those that are non-differentiable, and non-convex. GA's fall under the larger umbrella of evolutionary algorithms, and as the name suggests, they take inspiration from evolutionary processes to solve optimization problems. The specific GA used for this problem was the Non-Dominated Sorting Algorithm 2 (NSGA-II) which uses the following procedure to find the generate a Pareto front [2], [3], [5], [11] .

1) Random initialization of population of individuals with chromosomes (vector of decision variables).
2) Evaluate the fitness (performance) of each individual based on the objective functions.
3) Sort the population into different non-domination fronts. The first front is the set of individuals that are not dominated by any other individuals. The second front is the set of individuals only dominated by individuals in the first front, and so on.
4) Calculate the normalized distance between the neighboring individuals:
$$d_i = \frac{f_c(x_{i+1}) - f_c(x_{i-1})}{f_{c,\max} - f_{c,\min}} + \frac{f_{em}(x_{i+1}) - f_{em}(x_{i-1})}{f_{em,\max} - f_{em,\min}}$$
$f(x_{i+1})$ and $f(x_{i-1})$ are the objective values of the neighboring individuals. $f_{\max}$ and $f_{\min}$ are the minimum and maximum values of the objectives in that front. The boundary individuals are assigned $d_i = \infty$. Larger crowding distances are preferred over lower distances to promote diversity.
5) Use tournament selection to choose which individuals to progress to the next generation. Randomly select individuals $x^{(a)}, x^{(b)}$ and determine fitness based which is in a lower non-domination level. If $x^{(a)}, x^{(b)}$ are on the same front, then select the individual with the larger crowding distance.
6) Crossover the fit individuals by combining pairs of two 'parents' to create two 'children' with variables that fall on a probability distribution between the two parents.
7) Mutate the children by applying a small perturbation: $x_i + \delta \rightarrow x_i$.
8) Iterate steps 2-7 until convergence.

E. *MOEA/D*

In addition to the NSGA-II genetic algorithm, the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) was applied to the problem in an effort to improve upon the traditional genetic algorithm, NSGA-II. In MOEA/D the multiobjective problem is decomposed into many scalar optimization subproblems. In our case, we used a weighted sum to scalarize the multiobjective problem because it dovetailed nicely with the weighted sum algorithm used above. The algorithm starts by initializing a population, similar to the genetic algorithm framework. Each subproblem is then optimized using evolutionary operators such as tournament selection, crossover, and mutation. Unlike NSGA-II, however, each subproblem considers a defined number of neighboring subproblems—determined by Euclidean proximity in the weight space—and uses information from those neighbors to guide search and update solutions. This localized cooperation enables the population to progressively evolve toward a well-distributed Pareto front.

IV. Coding & Methods

A. *Programming Framework*

Advanced optimization algorithms and scalarization methods were implemented using the `pymoo` framework [13]. `OpenMDAO` was employed for problem formulation and implementation of the physics-based energy system models [14].The epsilon-constraint and weighted sum methods were implemented using SciPy's Sequential Least Squares Quadratic Programming (SLSQP) solver, which is capable of handling smooth non-linear constraints and bounds efficiently [15], [16]. Core numerical computations were performed using `NumPy` [17], and scientific routines were handled with `SciPy` [18]. `Pandas` provided tabular data structures [19], and visualizations were rendered using `Matplotlib` [20] and enhanced with `Seaborn` for statistical graphics [21].For machine learning routines and performance metrics, `scikit-learn` [22] was used, often with parallel processing support from `Joblib` [23]. Additional utilities for combinatorics and iteration came from the built-in `itertools` module [24]. To ensure differentiability and improve convergence in gradient-based optimization methods, standard non-differentiable functions such as $\max$, $\min$, and ReLU-like thresholds were replaced with smooth counterparts using the smoothing functions LogSumExp and Softplus [25], [26].

B. *Data Collection*

Real-world solar and wind data for the Navajo Nation was obtained through NREL's databases by using longitude and latitude. NREL's National Solar Resource Database (NSRDB) provided wind speed (m/s) at 80-meter hub height, and solar variables including global horizontal irradiance (GHI), diffuse horizontal irradiance (DHI), solar zenith angle (SZA), and azimuth angle at longitude -109.0430 and latitude 35.6778 for power calculations [27]. As local demand was not available for the existing Navajo Nation, the demand was estimated by scaling the kW/person demand from Tucson, AZ to the Navajo Nation [28].

C. *Feasibility of Solutions & Benchmarking*

A Monte Carlo variant of an exhaustive search algorithm was first implemented to systematically evaluate the objective function over a predefined discrete grid of points in the decision variable space. This stochastic method randomly sampled system configurations from the 4D decision space and gave us initial insight into the landscape of feasible and infeasible solutions. Although inherently less exhaustive, the Monte Carlo approach acted as a stochastic lower bound. If a multi-objective optimizer failed to outperform Monte Carlo solutions, it raised concerns about convergence or local optima [11], [12]. Conversely, a significant improvement over Monte Carlo validates the optimizer's effectiveness and ability to exploit the design space efficiently. A full 4D grid search was then employed as an additional baseline to benchmark performance. By discretizing each decision variable into 10 levels, the method exhaustively explored the design space and resulted in 10,000 simulations. The process guarantees complete coverage of the feasible region (within discretization resolution) and the true Pareto front without the influence of algorithmic heuristics [2], [29]. Further exploration revealed important differences between grid-searches for smooth and non-smooth versions-the feasible set for the optimization problem without a soft min and soft max produced significantly lower objective function values. This observation revealed how problem formulation can bias the optimizer's ability to explore shape of the feasible region and suggested that algorithms not reliant on gradients may be better suited for this problem by yielding a more accurate and beneficial Pareto front. These grid search techniques also served as reference points against which the efficacy, convergence behavior, and solution diversity of tested algorithms could be evaluated against. [1], [3]–[5].

V. Implementation & Discussion

A. *Results*

Figure 6 shows the Pareto Front results for all algorithms tested on the smooth problem.

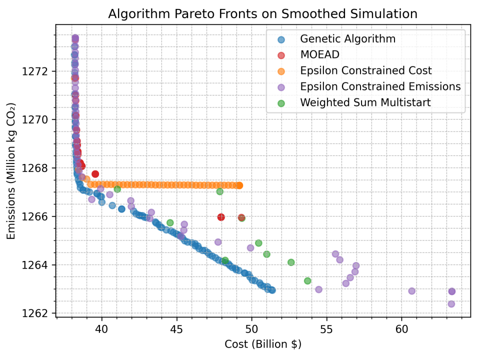


Fig. 6: Pareto front comparison for smooth problem.

Table I shows the runtime to find 20 Pareto front points by each of the algorithms for the smooth problem in hours:minutes:seconds (H:M:S).

Figure 7 shows the Pareto Front results for all algorithms tested on the non-smooth problem.

TABLE I: Algorithm runtime on smooth problem.

Algorithm	**Runtime (H:M:S)**
Epsilon Constraint (emissions constrained)	0:15:50
Epsilon Constraint (cost constrained)	0:15:52
Weighted Sum (singular initialization)	0:10:00
Weighted Sum (evolution inspired)	3:12:15
NGSA II	0:02:25
MOEA/D	0:03:25

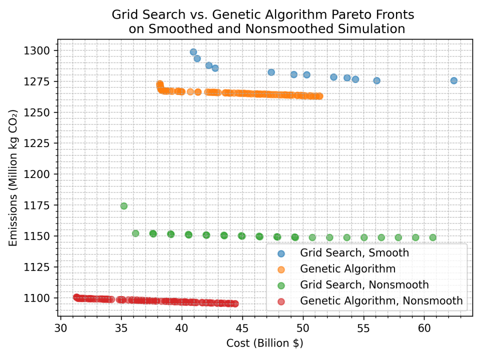
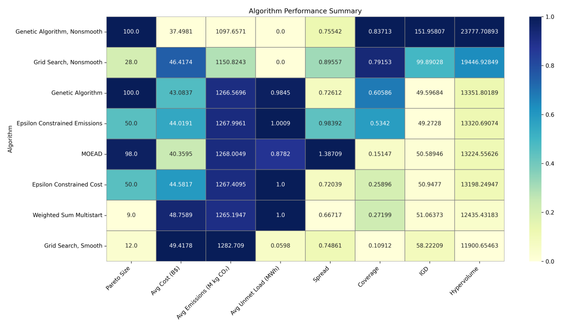


Fig. 7: Pareto Fronts of Grid Search and Genetic Algorithm on Smoothed vs. Non-smoothed Simulation.

A heatmap comparing performance metrics of algorithms is shown in **Figure 8**. Runtimes were compared using streamlined versions while preserving core logic without the computational overhead. Hypervolume (HV) measures provides an aggregate measure of convergence and diversity [2], [11]. Inverted Generational Distance (IGD) shows how closely an algorithm converges to the true front [11], [29]. Spread, the diversity metric Δ, assesses the uniformity and extent of distribution of solutions across the Pareto front [29].Coverage evaluates the frequency for one algorithm's solutions to dominate another's solution, indicating comparative performance [4], [5]. Average Cost, Average Emissions, and Average Unmet Load offer insight into the economic, environmental, and reliability performance of each algorithm's output set [1], [3]. Pareto Set Size quantifies the number of non-dominated solutions indicative of solution diversity flexibility [11].

The best performing algorithm was the genetic algorithm



	Pareto Size	Avg Cost (B$)	Avg Emissions (M kg CO₂)	Avg Unmet Load (MWh)	Spread	Coverage	IGD	Hypervolume
Genetic Algorithm, Nonsmooth	100.0	37.4981	1097.6571	0.0	0.75542	0.83713	151.95807	23777.70893
Grid Search, Nonsmooth	28.0	46.4174	1150.8243	0.0	0.89557	0.79153	99.89028	19446.92849
Genetic Algorithm	100.0	43.0837	1266.5696	0.9845	0.72612	0.60586	49.59684	13351.80189
Epsilon Constrained Emissions	50.0	44.0191	1267.9961	1.0009	0.98392	0.5342	49.2728	13320.69074
MOEAD	98.0	40.3595	1268.0049	0.8782	1.38709	0.15147	50.58946	13224.55626
Epsilon Constrained Cost	50.0	44.5817	1267.4095	1.0	0.72039	0.25896	50.9477	13198.24947
Weighted Sum Multistart	9.0	48.7589	1265.1947	1.0	0.66717	0.27199	51.06373	12435.43183
Grid Search, Smooth	12.0	49.4178	1282.709	0.0598	0.74861	0.10912	58.22209	11900.65463

Fig. 8: Algorithm performance heatmap.

outperforming all other algorithms for all metrics and was able to find the optimum Pareto front. The epsilon constraint method was able to find a Pareto front that was close to the optimum along the bottom edge of the Pareto front but far from the optimum on the left edge. The weighted sum was able to locate some points along the Pareto front, but was sensitive to initialization. The grid search performed the worst and was unable to find points along the Pareto front.

The SLSQP algorithm is capable of handling smooth nonlinear constraints and bounds efficiently and was used for both the weighted sum and epsilon constraint methods. At a high level, the SLSQP algorithm solves a least-squares approximation for the primal problem and then takes a newton step to move to the next iterate [15], [16], [30]. Since SLSQP is using, a Newton step, it is sensitive to gradient information. The microgrid optimization implementation uses a finite difference approximation for the gradient, and the implementation of SLSQP implementation uses first-order information to approximate the Hessian. These layered approximations and dependence on gradient information starkly contrasts with the operation of the evolutionary algorithms. The evolutionary algorithms are derivative-free optimizers, and thus not sensitive to gradient information nor descent directions of the objective function. These differences could explain why the evolutionary algorithms out performs the classical method as it directly attacks the optimization problem without approximations.

Algorithm operation leads to differences in computational complexity and what factors affect the runtime. The evolutionary algorithms used (MOEA/D and NGSA2) have iterations that are less computationally complex than iterations of epsilon constraint or weighted sum. Since SLSQP is used in the scalarization methods, each iteration solves a constrained least-squares optimization problem which is more computationally intensive than the formation of the next generation. However, there are nuances as genetic and classical are scaled. Increasing the input dimension increases the size of the least-squares optimization solved by SLSQP at each iteration of the scalarization methods and further increasing runtime of the scalarization. However, in certain multi-objective problems, evaluating the objective functions are computationally expensive. Genetic algorithm iterations with large populations would have large runtime because of evaluating the fitness of every individual.

During experimentation, the weighted sum Pareto front was found to be sensitive to initialization likely due to a local minimum. The epsilon constraint method uses the same single variable optimization algorithm as weighted sum, but the iterations of the single variable are distinct. At each iteration the epsilon constraint method has both the unmet load and an objective function as constraints whereas weighted sum only has the constraint of unmet load. The difference in constraints results in a different lagrangian at each step which could explain why weighted sum is caught by a local minimum and epsilon constraint is not. To work around the sensitivity, the weighted sum was implemented with an evolutionary inspired iteration. For each weight combination 5 random initializations are optimized with the best performing being the output at that weight combination. This implementation was much more successful and sampling points along the full Pareto front but the increased computational complexity significantly increased the runtime.

B. *Key Findings*

The Genetic Algorithm had the strongest performance due to its low runtime and computational intensity. The Genetic Algorithm also was the most flexible algorithm for handling the complexities required to optimize a non-convex, non-smooth optimization problem. The results underscore the importance of algorithm selection and problem formulation in multi-objective optimization. The derivative-free GA proved especially effective on this non-smooth problem, suggesting robustness to non-differentiable dynamics and discontinuities. In contrast, classical methods relying on gradient information struggled to match this performance, highlighting potential limitations when applied to complex energy systems with implicit, non-smooth constraints.

C. *Limitations & Challenges*

A key challenge was the trade-off between problem smoothness and optimizer compatibility: while smoothing enabled gradient-based methods, it also altered the feasible set and masked key system dynamics. Additionally, many optimizers were sensitive to initialization and local minima, requiring multi-start or hybrid strategies to improve coverage. Computational cost also limited the resolution of exhaustive methods like grid search, constraining their ability to explore higher-dimensional design spaces.

VI. Conclusion

This model is a strong starting framework for any off-grid area that wants to minimize cost and greenhouse gas emissions when deciding the power generation sources for a given area and demand curve. Potential changes that could be made to improve the scalability of this project include adding other generation technologies, scaling the emissions during operation and maintenance for each technology, adding land area constraints for each technology for a maximum number of each generation source, and adding specific models and manufacturers for each generation technology such as wind turbines or solar panels for more accurate capacity factors and costs. Other generation sources could include additional generation types such as natural gas power plants, hydro-powered dams, geothermal, nuclear reactors, and more, depending on the location of the off-grid model. This model does not include any costs or constraints for transmission lines, substations, or safety factors for grid protection (too much or too little power back to the grid could create a problem). Other than scaling the model for other specific locations and generation types as mentioned previously, the model could be further developed using other SCIPY single objective optimizers in place of MDAO.

References

[1] B. Zhao, X. Zhang, P. Li, K. Wang, M. Xue, and C. Wang, "Optimal sizing, operating strategy and operational experience of a stand-alone microgrid on dongfushan island," *Applied Energy*, vol. 113, pp. 1656–1666, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261913007629

[2] R. Dufo-López and J. L. Bernal-Agustín, "Multi-objective design of pv–wind–diesel–hydrogen–battery systems," *Renewable Energy*, vol. 33, no. 12, pp. 2559–2572, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148108000724

[3] H.-C. Chen, "Optimum capacity determination of stand-alone hybrid generation system considering cost and reliability," *Applied Energy*, vol. 103, pp. 155–164, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261912006630

[4] B. Y. Ekren and O. Ekren, "Simulation based size optimization of a pv/wind hybrid energy conversion system with battery storage under various load and auxiliary energy conditions," *Applied Energy*, vol. 86, no. 9, pp. 1387–1394, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261908003401

[5] H. Jahangir, A. Ahmadian, and M. A. Golkar, "Multi-objective sizing of grid-connected micro-grid using pareto front solutions," in *2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, 2015, pp. 1–6.

[6] L. Zhu, Z. Yu, N. Liu, and J. Xu, "Whole life cycle optimal allocation of the energy storage systems in a distributed network," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 5556–5561.

[7] National Renewable Energy Laboratory (NREL), "2024 Annual Technology Baseline: Utility-Scale PV," 2024, [Online; accessed 29-April-2025]. [Online]. Available: https://atb.nrel.gov/electricity/2024/utility-scale_pv

[8] ——, "2024 Annual Technology Baseline: Land-Based Wind," 2024, [Online; accessed 29-April-2025]. [Online]. Available: https://atb.nrel.gov/electricity/2024/land-based_wind

[9] ——, "2024 Annual Technology Baseline: Utility-Scale Battery Storage," 2024, [Online; accessed 29-April-2025]. [Online]. Available: https://atb.nrel.gov/electricity/2024/utility-scale_battery_storage

[10] U.S. Department of Energy, "How wind can help us breathe easier," 2023, accessed: April 29, 2025. [Online]. Available: https://www.energy.gov/eere/wind/articles/how-wind-can-help-us-breathe-easier

[11] *Multiobjective Optimization*. London: Springer London, 2010, pp. 193–262. [Online]. Available: https://doi.org/10.1007/978-1-84996-129-5_6

[12] Y. Cheng, Y. Jin, and J. Hu, "Adaptive epsilon non-dominated sorting multi-objective evolutionary optimization and its application in shortest path problem," in *2009 ICCAS-SICE*, 2009, pp. 2545–2549.

[13] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[14] J. S. Gray, K. T. Moore, B. A. Naylor, T. A. Hearn, P. S. Beran, and J. R. Martins, "Openmdao: An open-source framework for multidisciplinary design, analysis and optimization," in *10th AIAA Multidisciplinary Analysis and Optimization Conference*. AIAA, 2010.

[15] SciPy developers, "scipy.optimize.minimize — slsqp (sequential least squares programming)," https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html, 2024, accessed: April 29, 2025.

[16] Wikipedia contributors, "Sequential quadratic programming," https://en.wikipedia.org/wiki/Sequential_quadratic_programming, 2024, accessed: April 29, 2025.

[17] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[18] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[19] J. Reback, W. McKinney, jbrockmendel, J. Van Den Bossche, T. Augspurger, P. Cloud, S. Hawkins, Gfyoung, A. Klein, T. Petersen *et al.*, "pandas-dev/pandas: Pandas," *Zenodo*, 2020.

[20] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[21] M. L. Waskom, "Seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, pp. 2825–2830, 2011.

[23] J. Developers, "Joblib: running python functions as pipeline jobs," 2024, [Online; accessed 29-April-2025]. [Online]. Available: https://joblib.readthedocs.io

[24] Python Software Foundation, "itertools — functions creating iterators for efficient looping," 2024, accessed: April 29, 2025. [Online]. Available: https://docs.python.org/3/library/itertools.html

[25] Wikipedia contributors, "Logsumexp," https://en.wikipedia.org/wiki/LogSumExp, 2024, accessed: April 29, 2025.

[26] ——, "Softplus," https://en.wikipedia.org/wiki/Softplus, 2024, accessed: April 29, 2025.

[27] National Renewable Energy Laboratory, "National solar radiation database (nsrdb) data viewer," 2025, accessed: May 7, 2025. [Online]. Available: https://nsrdb.nrel.gov/data-viewer

[28] U.S. Energy Information Administration, "Hourly electric grid monitor – u.s. overview (us48)," 2025, accessed: May 7, 2025. [Online]. Available: https://www.eia.gov/electricity/gridmonitor/dashboard/electric_overview/US48/US48

[29] S. Smith, M. Southerby, S. Setiniyaz, R. Apsimon, and G. Burt, "Multiobjective optimization and pareto front visualization techniques applied to normal conducting rf accelerating structures," *Phys. Rev. Accel. Beams*, vol. 25, p. 062002, Jun 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevAccelBeams.25.062002

[30] D. Kraft, *A Software Package for Sequential Quadratic Programming*, ser. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. [Online]. Available: https://books.google.com/books?id=4rKaGwAACAAJ

1

Appendix 1

Battery SOC update: $SOC[t{+}1] = SOC[t] + \eta P_{ch}[t] - \frac{P_{dis}[t]}{\eta}$

SOC range: $0.2\,x_b \leq SOC[t] \leq x_b$

Charge/discharge limits: $0 \leq P_{ch}[t],\ P_{dis}[t] \leq C_{rate} \cdot x_b$

Load balance: $P_s[t] + P_w[t] + P_d[t] + P_{dis}[t] \geq L_t.$

Diesel min loading: $0.3\,x_d \leq P_d[t] \leq x_d$

Initial SOC: $SOC[0] = 0.5\,x_b.$

Solar output: $P_{\mathrm{s},t}(x_s) = \eta_{\mathrm{s}} A_{\mathrm{panel}}(x_s) I_t^{\mathrm{eff}}$

Wind output: $P_{\mathrm{w},t}(x_w) = \begin{cases} \min\left(\frac{1}{2}\rho_w A C_p v_t^3,\ x_w\right), & v_{\text{cut-in}} \leq v_t \leq v_{\text{rated}} \\ x_w, & v_t > v_{\text{rated}} \wedge v_t \leq v_{\text{cut-out}} \\ 0, & \text{otherwise} \end{cases}$

Power non-negativity: $P_s[t],\ P_w[t],\ P_d[t],\ P_{ch}[t],\ P_{dis}[t] \geq 0$