Constrained Evolutionary Wind Turbine Placement with Penalty Functions

Daniel Lückehe
Department of Geoinformation
Jade University of Applied Sciences
Oldenburg, Germany
daniel.lueckehe@uni-oldenburg.de

Markus Wagner
Optimisation and Logistics
University of Adelaide
Adelaide, Australia
markus.wagner@adelaide.edu.au

Oliver Kramer
Department of Computing Science
University of Oldenburg
Oldenburg, Germany
oliver.kramer@uni-oldenburg.de

***Abstract*—Geographical constraints are essential when planning the locations for wind turbines. In real-world scenarios, especially in densely populated countries, the designated area where turbines can be placed is not an empty map on which the turbines can be placed arbitrarily. Even in rural areas, streets, buildings, and rivers have to be considered. In this paper, we model two constrained turbine placement scenarios and use evolutionary algorithms to find optimized turbine locations. To evaluate the locations, we combine a proven wind model with real-world data of a wind prediction model from a meteorological service. Geographical data from a free map service is used to define constrained areas in the scenarios based on administrative rules. For the evolutionary optimization process, we consider five ways to handle penalties. Starting with a simple specification that can only achieve two different values, we end up in a definition that considers distances relative of the required minimum distances to all geographical objects for each turbine. We combine the penalty definitions with three types of penalty functions. In the experimental section, we compare the various configurations and show a detailed analysis of the results.**

I. Introduction

The reduction of greenhouse gas emissions to slow down the climate change is an important current challenge. It is supported by many participants, e.g., the 2° goal was agreed upon at the 2015 United Nations Climate Change Conference in Paris [18]. To reduce the greenhouse gas emissions, a lot of steps have to be taken. One of them is the integration of renewable sources into our energy supply. Unlike conventional power plants, the behavior of renewable energy sources like photovoltaic plants and wind turbines is strongly depending on their location, e.g., the energy output of a wind turbine depends on the wind potential at its location. In this paper, we are focusing on the geographical planning of wind turbines. The determination of optimal locations that are suitable for turbines is called wind turbine placement optimization. In real-world scenarios of densely populated countries, there are no empty maps, thus, the environment always has to be taken into account. We need to consider the geographical constraints in order to create feasible solutions that are useable in real-world. Besides the consideration of the constraints, the optimization of the power output is important to increase the competitiveness of wind farms in the energy market.

In this paper, we use a wind model based on the proven model of Kusiak and Song [10] and data of a wind prediction model from a meteorological service to evaluate locations of the turbines.[1] We propose five ways how to compute the penalties G that are used by three variants of penalty functions to handle the constrained wind turbine optimization problem. To analyze the behavior of the various combinations of the penalty G and the penalty functions, we specify two constrained turbine placement scenarios that are based on the data of a free topological map service. In these scenarios, we define minimum distances to buildings, streets, and rivers taken from administrative rules. The scenarios are treated as black-box optimization problems and solved by an evolutionary algorithm using an adaptive step-size control.

This paper is structured as follows. In Section II, an overview over the related work is given, followed by the introduction of our wind model in Section III. The used geographical data are described in Section IV. In this section, we also specify the constrained turbine placement scenarios and definitions of penalty G. In Section V, the optimization approach including the evolutionary algorithm and the penalty functions are explained. We present our experimental investigations in Section VI and draw conclusions in Section VII.

II. Related Work

In the following, we outline some of the relevant works. Our model is based on the work by Kusiak and Song [10], which uses wind distributions to compute the power output of a wind farm. Their approach applies Jensen's wake model [14], which allows very efficient computations of wake effects. They use a simple evolution strategy [1] to optimize very small wind farms. The more complex CMA-ES has been used in [20] for the effective optimization of up to 1000 turbines on empty maps. There, layout boundaries and intra-turbine distance constraints were considered rather inefficiently by a death-penalty for the infeasible solutions, i.e., new layouts were re-sampled until a feasible layout was generated. In [19] a random local search was presented that combines a problem-specific operator with an asymptotic speed-up of the computation time of the wake effects. In our recent article [11], we present a correction to the commonly used wake model by Kusiak and Song [10]. Further, we consider geographic real-world constraints that are based on OpenStreetMap data and real meteorological data. Also here, constraints were considered using the death-penalty approach.

[1]The model is available at: http://geo-planning.tumblr.com.

While heuristic optimizers are commonly used to solve real-world problems, they encounter difficulties in solving them when they include non-trivial constraints [12]. Over the past decades, a plethora of constraint handling techniques has been developed. The variety of methods range from penalty functions that decrease the fitness of infeasible solutions [5] and decoder functions that let the search take place in another unconstrained or less constrained solution space [7] to feasibility preserving approaches that adapt representations or operators [15]. Multi-objective approaches that treat each constraint as an objective also have been investigated [2]. For this sake, evolutionary multi-objective optimization methods like NSGA-II can be adapted. Penalty functions are simple and powerful methods to handle constraints. A simple form is death penalty that rejects infeasible solutions and generates new ones until a sufficient number of feasible candidates have been generated. For a more extensive survey on constraint handing for EAs, see [8].

III. Wind Model

In this section, our wind model is introduced. It is used as fitness function f to assess wind farm layouts. The model computes the power output E of multiple wind turbines that are described in a solution $\mathbf{x}$. The solution $\mathbf{x}$ is a vector of elements $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ with the length N. It codes the x- and y-coordinate of the turbines which we specify for a single turbine i as $\mathbf{t}_i = (x_i^t, y_i^t)$. We also define $N_t = N/2$. Thus, the solution vector $\mathbf{x}$ can be written like follows:

$$\mathbf{x} = (x_1^t, y_1^t, x_2^t, y_2^t, \ldots, x_{N/2}^t, y_{N/2}^t) = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{N_t}).$$

To compute the energy output $E(\mathbf{x})$ of multiple turbines, we use the power curve of a real wind turbine by Enercon, data from a numerical wind vector prediction model from the German Weather Service, and a power calculation based on a model from Kusiak and Song [10].

A. Wind Turbine

In this paper, the wind turbine power curve β is based on the parameters of the Enercon E92, which is a modern and versatile turbine. We use a height of 78 m. The points of the power curve from the manufacturer's data sheet are shown in Figure 1(a). If the wind speed v is lower than the cut-in

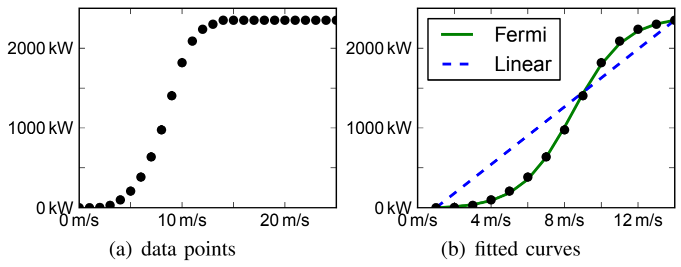


Fig. 1. Power curve of wind turbine E92

speed $v_{\text{cut-in}}$, the turbine produces no energy. For wind speeds higher than the rated v_{rated}, the turbine produces P_{rated}. The plot shows a development between 1 m/s to 14 m/s that is first convex changing to a concave shape. As the Fermi function has the same properties and is monotonically increasing, we fit this function – instead of a linear one $(\lambda \cdot v + \eta)$, which is commonly used in the wind farm optimization literature.

$$\beta(v) = \begin{cases} 0 & v < v_{\text{cut-in}} \\ \frac{1}{1+e^{-k\cdot(v-\mu)}} \cdot m + d & v_{\text{cut-in}} \le v < v_{\text{rated}} \\ P_{\text{rated}} & v_{\text{rated}} \le v \end{cases} \quad (1)$$

Equation 1 shows the wind turbine power curve. For the E92 the following technical parameters apply

$$k \approx 0.705, \mu \approx 8.430, m \approx 2409.336, d \approx -12.735.$$

The comparison of the Fermi- and the linear function is visualized in Figure 1(b). It can be seen that the Fermi function matches a lot better.

B. Numerical Wind Vector Prediction Model

To predict the wind potential of a position, we use wind vectors from the COSMO-DE analysis data. These data are provided from the German Weather Service [3]. On a grid over Germany with nearly 200 000 points, there are hourly wind vectors for every grid point. We are using the data from 2012 which lead to more than 8500 vectors per grid point. These wind vectors are collected depending on their speed and angle. We use a resolution for the wind speed of 0.3 m/s and 64 different wind angle as this configuration leads to only small discretization errors while keeping the computational complexity manageable.

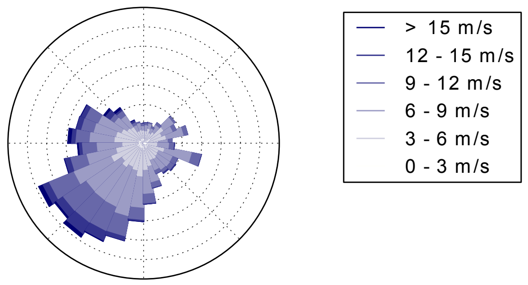


Fig. 2. Example of a windrose

Figure 2 shows a wind rose that visualizes the collected wind vectors. It shows from where the wind is blowing, the frequency represented by the length of the bars, and the wind speed visualized by color. For a position between the grid points of the COSMO-DE model, we interpolate the data using inverse distance weighting of the nearest nine points.

C. Power Calculation

The used model to calculate the power output E_t from a wind turbine $\mathbf{t}$ over a period of time is based on the approach from Kusiak and Song [10]. The basic idea in this model is to split the wind rose into multiple pieces and describe the wind of every piece as Weibull distributions [21]. By multiplying these distributions with a linear wind turbine power curve, we get the produced power. Unfortunately, this quite nice approach leads to integrals which are not solvable in a pure analytical way. Thus, parts of the integrals are solved by using the Riemann sum which leads to a discretization of the wind speed v and the wind direction θ. Due to this discretization, inhomogeneous structures arise in the solution space. This behavior is analyzed and solved in [11]. To make it clear that parts of the power output are calculated by summing, we will specify the power calculation directly as a sum in this paper. For one wind direction θ, it applies:

$$E_{t,\theta}(\mathbf{t}_i, \theta) = \sum_{j=0}^{\infty} \beta_i(v_j) \cdot \omega(v_j, \mathbf{t}_i, \theta). \qquad (2)$$

Theoretically, the sum goes to infinity, however, in practice it goes from the cut-in speed of the turbine to the wind speed when the turbine turns off for safety reasons. The variable ω describes the occurrence of a wind speed depending on its position and its wind angle. It is normalized to the sum of all wind vectors. The wind speeds v_j result from the discretization, e.g., in our work v_1 represents $0.3\,\mathrm{m/s}$ and v_4 stands for $1.2\,\mathrm{m/s}$. Due to wake effects which cause a reduction of the wind speed behind, w. r. t. the wind direction, a placed turbine v_j can be reduced. To compute these effects, we use the Jensen's wake model [14]. The overall output of a turbine can be specified as follows:

$$E_t(\mathbf{t}_i) = \sum_{k=0}^{2\pi/\theta_N} E_{t,\theta}(\mathbf{t}_i, \theta_k), \qquad (3)$$

where θ_N is the number of sampling steps over the wind angle θ in radians. Besides the advantage of making the discretization explicit, the formulation can be used without fitting the Weibull distribution employing raw wind data. Instead of fitting the Weibull distribution to the data and sampling from the fitted model, we employ the data directly and thus avoid a bias of the model. Another benefit from this description is that all parts are solved by summing up. Hence, the power curve can be replaced easily. This makes it possible to substitute the originally linear power curve with our Fermi function based power curve from Section III-A. The produced energy of all turbines E is the sum of the produced energy E_t of each turbine $\mathbf{t}$ existing in the solution vector $\mathbf{x}$, resulting in the fitness function:

$$f(\mathbf{x}) = \sum_{i=1}^{N_t} E_t(\mathbf{t}_i) \qquad (4)$$

that is basis of our evolutionary optimization process employing $\mathbf{x} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{N_t})$.

IV. Geographical Data

In densely populated countries like Germany, it is not possible to place turbines on completely empty maps. The placements must be integrated into the environment and minimum distances have to be kept. This problem not only arises when wind turbines are to be placed next to cities, but even in rural areas there are streets, buildings, and rivers.

In this paper, we use the minimum distances that have to be kept for defining the penalty G of the optimization problem. As source for the geographical data, we use the data from a map service. After specifying rules for minimum distances, we introduce two constrained turbine placement scenarios. Both scenarios comprise thousands of constraints. Each constraint belongs to a group, e.g., residential buildings. This allows different definitions of penalty G.

A. *OpenStreetMap*

As map service, we use OpenStreetMap (OSM), which is a community-driven project. For countries, in which OSM is well developed, its geographical data are becoming comparable in quality to commercial providers [13]. We use the geographical data to get the locations of residential buildings, non-residential buildings, big streets like highways, small streets like secondary roads, and rivers. To model minimum distances around buildings a simple circle is required. One building leads to one constraint. Streets and rivers are represented by many straight lines in the data structure of OSM. For every line, we use two circles and one rectangle to model the minimum distances around it.

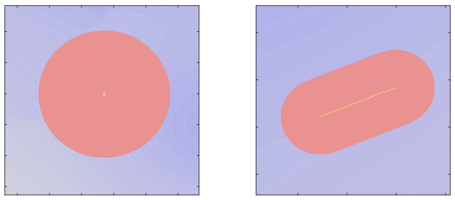


Fig. 3. One building (gray) and one street (yellow) surrounded by constrained red areas. In the background the structure of the wind potential at position is shown.

Figure 3 shows the basic elements for buildings and streets / rivers. On the left side of the plot, a gray building is shown. It is surrounded by a red circle that visualizes the infeasible area around the building. In the background, the potential map based on the wind model is shown. On the right hand side, we can observe a yellow part representing a street. This part is also surrounded by a red area visualizing the infeasible space. Again, we can see the potential map based on the wind model in the background.

B. *Minimum Distance Rules*

In real-world turbine placement, there are a lot of different minimum distances to geographical objects that have to be considered. Thereby, one problem is the missing of general rules or laws for the size of the distances, e.g., in Germany, there are no rules for the whole country but for the individual federal states. Unfortunately, the federal states have different rules. In this paper, we use distances oriented to the case study of Höfer *et al.* [4]. For residential buildings, we use the 10H-rule applied in Bavaria leading to larger distances. For streets, we follow the legal requirements as defined by the German Federal Highway Act. In practice, exceptions for this rules are possible. Table I shows an overview over all used constraint types.

TABLE I. Minimum distances to wind turbines

Object	Distance
Residential building (10H-rule with turbine height 78 m)	780 m
Non-residential building	400 m
Big street (e.g. highways)	100 m
Small street (e.g. secondary roads)	40 m
River	50 m

C. *Scenarios*

Based on the OpenStreetMap data, we define two constrained turbine placement scenarios. In both cases the outer dimensions are $5\,\mathrm{km} \times 5\,\mathrm{km}$. To consider geographical objects even at the borders of the scenarios, only the inner $4\,\mathrm{km} \times 4\,\mathrm{km}$ can be used for the placement of turbines. We place 30 turbines in each scenario. This wind farm size is used as most wind farms in Lower Saxony consist of 30 or fewer wind turbines [16]. For wind farms at this scale even small improvements in their power output lead to significant income increases, e.g., a wind farm with 25 turbines like the Enercon E92 with $2.35\,\mathrm{MW}$ and a load of 20% of full load hours produces more than $100\,\mathrm{GWh}$ per year.

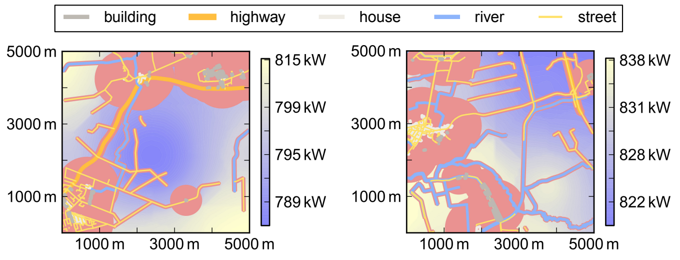


Fig. 4. Scenarios 1 (left) and Scenario 2 (right)

Figure 4 illustrates Scenario 1 and 2. On the left hand, we can see Scenario 1. It lies at the area in decimal degrees from 53.41077° to 53.45648° and from 7.74448° to 7.81844°. Scenario 2, that is shown on the right part of Figure 4, is located at the region from 53.3394° to 53.38524° and from 7.5182° to 7.591795°. On the plots, the residential buildings are shown in white, non-residential buildings are visualized in gray. Orange lines stand for large streets and small streets are represented in yellow. Blue lines show the locations of rivers. The objects are surrounded by red areas that visualize the constrained space based on the minimum distances that has to be maintained. In the background the potential map based on the wind model is shown in varying shades from blue to light yellow.

TABLE II. NUMBER OF CONSTRAINTS IN THE SCENARIOS

Kind	Residential Buildings	Non Residential Buildings	Small Streets	Large Streets	Rivers
Scenario 1					
Objects	112	117	140	5	21
Parts	112	117	771	109	192
Constraints	112	117	2313	327	576
Scenario 2					
Objects	244	24	184	10	63
Parts	244	24	1114	31	759
Constraints	244	24	3342	93	2277

Table II presents the number of constraints in both scenarios. To summarize, Scenario 1 consists of $G_N = 3445$ constraints and Scenario 2 is composed of $G_N = 5980$ constraints.

D. Penalty

In this paper, we define five different specifications of how to calculate the penalty G. The specifications have an increasing complexity. In the simplest approach, the penalty G gets the value $G(\mathbf{x}) = 0$ for a feasible solution $\mathbf{x}$ and it applies $G(\mathbf{x}) = 1$ in every other case. We call this penalty function $G_{[1 \in 01]}$. The labels of the penalties G follow the notation: [maximum value $\in$ co-domain]. The second function $G_{[N_t \in \mathbb{N}_0]}$ counts the turbines that violate any constraint. The third function $G_{[G_N \in \mathbb{N}_0]}$ counts for every turbines the number of constraint violations. The fourth penalty function $G_{[N_t \in \mathbb{R}]}$ is like the second one but it uses a continuous co-domain. This means that it can differentiate between, e.g., turbines that violate the building vicinity constraint depending on how close to the building they actually are. The last penalty function $G_{[G_N \in \mathbb{R}]}$ is based on the third function but uses a continuous co-domain like the fourth function.

To make the differences between the penalty specifications clear, we describe three examples and show the matching values of G in Table III.

TABLE III. PENALTY FUNCTIONS

Function	Max. Value	Co-domain	Example 1	Example 2	Example 3
$G_{[1 \in 01]}$	1	$\{0, 1\}$	1	1	1
$G_{[N_t \in \mathbb{N}_0]}$	N_t	$\mathbb{N}_0$	1	1	2
$G_{[G_N \in \mathbb{N}_0]}$	G_N	$\mathbb{N}_0$	2	5	5
$G_{[N_t \in \mathbb{R}]}$	N_t	$\mathbb{R}$	0.6	0.6	0.9
$G_{[G_N \in \mathbb{R}]}$	G_N	$\mathbb{R}$	0.8	1.5	1.5

In the examples the following constraints are violated: in Example 1, turbine 1 violates two residential buildings (60%, 20%). In Example 2, turbine 1 violates two residential buildings (60%, 20%) and three small streets (30%, 30%, 10%). While in Example 3, turbine 1 violates two residential buildings (60%, 20%) and turbine 2 violates three small streets (30%, 30%, 10%). The percentages represent the distances to the objects, e.g., 90% for a non-residential building result from a turbine only $40\,\mathrm{m}$ away from the building, while in the case of 10% the turbine is only $40\,\mathrm{m}$ away from the feasible space. So, it is $360\,\mathrm{m}$ away from the building.

Besides the values for G in the examples, Table III shows the co-domain and maximum value of each penalty specification. The variable N_t describes the number of turbines as specified in Section III and G_N is the count of constraints. Its concrete values are shown in Section IV-C. Figure 5 shows how a placed turbine would change the value of the penalty G for the different variants. In this view, there is no difference between $G_{[1 \in 01]}$ and $G_{[N_t \in \mathbb{N}_0]}$ as only one turbine would be placed. This equality is also shown in Table III by Example 1 and Example 2. The table shows that $G_{[1 \in 01]}$ and $G_{[N_t \in \mathbb{N}_0]}$ are different for multiple turbines.

V. OPTIMIZATION

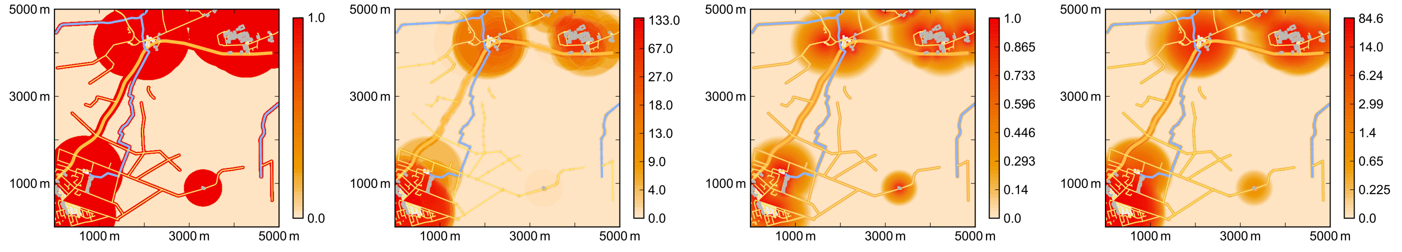
The turbine placement is treated as a black-box optimization problem with the objective to maximize the function $\widetilde{f}(\mathbf{x})$ that depends on the fitness value computed by the wind model f and it depends on the penalty G based on the geographical data. The function $\widetilde{f}(\mathbf{x})$ is defined as follows:

$$\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) - \alpha \cdot G(\mathbf{x}) \qquad (5)$$

where the penalty factor α determines the influence of the penalty G. There are different approaches to control α, which we present in Section V-B.

A. Evolutionary Algorithms

To solve the turbine placement problem, we use randomized optimization approaches called evolutionary algorithms (EAs) based on a population p of candidate solutions. As the focus of this paper are the penalty G and the penalty



(a) $G_{[N_t \in \mathbb{N}_0]}$ (b) $G_{[G_N \in \mathbb{N}_0]}$ (c) $G_{[N_t \in \mathbb{R}]}$ (d) $G_{[G_N \in \mathbb{R}]}$

Fig. 5. The induced penalty G when placing one turbine on the map in Scenario 1 for the penalty variants $G_{[N_t \in \mathbb{N}_0]}, G_{[G_N \in \mathbb{N}_0]}, G_{[N_t \in \mathbb{R}]}$, and $G_{[G_N \in \mathbb{R}]}$. The non-linear colorbars with different scalings illustrate the resulting penalty functions G.

functions to control α, we focus on the application of only one EA. A comprehensive comparison of different EAs for the turbine placement problem by using a death penalty function can be found in [11]. In this paper, we apply an adaptive $(30+50)$-EA. Based on preliminary experiments this population size and selection pressure turn out to maintain sufficient diversity during the optimization process. In every generation a new population p' is created consisting of $\lambda = 50$ new solutions. The best $\mu = 30$ solutions, w. r. t. $\widetilde{f}$, are chosen from the new population p' and the population p of the last generation. When using a variable constraint handling method, i.e., the penalty factor α is changing during the optimization process, $\widetilde{f}$ has to be recomputed for all solutions from the last generation. While this seems to be expensive, for a solution $\mathbf{x}_i$ the fitness value $f(\mathbf{x}_i)$ and the penalty $G(\mathbf{x}_i)$ remain unchanged, so the recalculation of $\widetilde{f}(\mathbf{x}_i)$ after a change of α is very cheap when $f(\mathbf{x}_i)$ and $G(\mathbf{x}_i)$ have been saved in the last generation.

Variation: Turbine-oriented mutation operators are very powerful like shown in [11], i.e., the mutation randomly picks one turbine in each mutation step and moves the turbines based on the Gaussian distribution. No recombination is applied, but the EA randomly selects a solution $\mathbf{x}$ from the best μ solutions of the last generation as base for a new solution $\mathbf{x}'$ and generates a novel position of a turbine $\mathbf{t}_i'$ with the Gaussian operator:

$$\mathbf{t}_i' = \mathbf{t}_i + \sigma \cdot (\mathcal{N}(0,1), \mathcal{N}(0,1)) \,. \tag{6}$$

The step size σ is controlled adaptively. We are using the Rechenberg's step size control [1], i.e., if more than 20% of the mutation operations lead to an improvement, the step size will be increased and vice versa. A mutation is evaluated as an improvement, if the created new solution will be picked in the selection phase and thus improves the population p. As increasing / decreasing factor τ, we are using $\tau = 1.1$. The maximum step size is set to the usable map size of $4000\,\text{m}$.

Selection: The solutions with the highest $\widetilde{f}$ values are selected from population p' for the population p of the following generation. In case of a variable penalty function that changes the penalty factor α, the $\widetilde{f}$ values from p are recomputed.

B. Penalty Functions

The penalty function controls penalty factor α and thus determines how the optimization process treats a constraint violating solution $\mathbf{x}_i$, for which $G(\mathbf{x}_i) \neq 0$ holds. The acceptance of infeasible solutions during the optimization process allows the exploration of infeasible solution space areas in a constrained black-box optimization problem. This can be beneficial while searching for a worthwhile solution that lies on a feasible island in the solution space surrounded by constraints [8].

In this work, we are using three different types of penalty functions. In the first approach, the penalty function does not change penalty factor α during the optimization process. We call this approach *constant*. The second method controls α adaptively depending on the number of feasible solutions in the current population p. It is called *adaptive*. In our last *balanced* approach, α is adapted to balance the ratio of feasible and infeasible solutions. To summarize, we use a constant approach with a fixed α, an adaptive approach changing α relatively slow, and a balanced approach adapting α comparatively fast.

Constant: In the constant approach, the penalty factor α is set to $\alpha_c = 70500$ which corresponds to the maximum power that can be produced by all wind turbines in our settings with 30 turbines. If the penalty G of a solution $\mathbf{x}_i$ is $G(\mathbf{x}_i) = 1$, it will lead to a lower value of $\widetilde{f}$ than any feasible solution and force the optimization process to explore the feasible solution space.

Adaptive: The adaptive penalty function controls α as proposed by Kramer *et al.* [9]. It adapts the idea of the Rechenberg's step size control to the area of penalty functions as follows. If less than 20% of the current population is feasible, α is increased, otherwise it is decreased. In our approach, the penalty factor α is modified by 1.023, which allows changing α in 100 generations by a factor of 10. In experimental runs with 1000 generations, α can be modified by factor 10^{10} during the whole optimization process. As for few generations, factor 1.023 is relatively small, this approach behaves significantly different to the balanced approach. The function starts with $\alpha = \alpha_c$.

Balanced: The balanced approach adapts the penalty factor α, in order to enforce a target ratio of feasible and infeasible solutions. To find an adequate setting for α, a simple $(1+1)$-EA is used that adapts α with a multiplicative rule. Objective is to achieve an equal balance of feasibility in the population, i.e., $\mu/2$ feasible and $\mu/2$ infeasible solutions. In case of less than $\mu/2$ feasible solutions in the population, the penalty factor α is set to $\alpha = \alpha_c$. Hence, the optimization process focuses on creating feasible solutions. If less than $\mu/2$ of the solutions are infeasible, we set $\alpha = 0$ to give the process the possibility to more explore the solution space.

C. Initial Solution

We use two methods to create initial solutions. In the first approach, the initial solutions $(\mathbf{x}_{01}, \ldots, \mathbf{x}_{0\mu})$ are generated randomly. Due to the highly constrained solution space, the chance is extremely high that the created solutions are not feasible, i.e., $G(\mathbf{x}_{0i}) \neq 0$ for $i = (1, \ldots, \mu)$. Generating a feasible solution based on an infeasible one is a challenging task and a good test for the constraint mechanisms. We call this initialization method *random*. In the second approach, the turbines are also placed randomly on the map. But in the case that a turbine violates any constraints, the turbine is replaced by a new randomly chosen place. This is repeated until all turbines are located in feasible areas. The method generates random solutions $(\mathbf{x}_{01}, \ldots, \mathbf{x}_{0\mu})$, but applies $G(\mathbf{x}_{0i}) = 0$ for $i = (1, \ldots, \mu)$. This approach is referred to as *feasible* in the following.

VI. Experimental Results

In this section, we present our experimental results. All experiments run for 1000 generations with the setting presented in Section V and are repeated 100 times. The tables in this section show the mean and corresponding standard deviation of these experiments. In case of the adaptive penalty function, slight constraint violations may occur due to numerical reasons. In the range of $\epsilon = 0.001$, they are not relevant in practical applications. For example, a constraint violation of 0.001 corresponds to a deviation of 0.4 m for a non-residential building. We will treat final solutions as feasible, if a constraint violation below ϵ occurs and will indicate that special cases explicitly.

A. Comparison with Random Initialization

First, we analyze the results of the different penalty functions and kinds of penalty starting with a random initialization. Table IV shows the results. Thereby, P stands for the power which corresponds to the fitness function f.

TABLE IV. Experimental results of Scenario 1 using random initial solutions.

	Penalty Function		
	Constant	Adaptive	Balanced
Penalty	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW
$G_{[1\in 01]}$	not feasible	not feasible	not feasible
$G_{[N_t\in\mathbb{N}_0]}$	21 640.65 ± 34.41	not feasible	21 637.00 ± 44.90
$G_{[G_N\in\mathbb{N}_0]}$	21 649.94 ± 39.14	most not feasible	21 697.09 ± 41.89
$G_{[N_t\in\mathbb{R}]}$	21 659.17 ± 38.55	21 679.19 ± 47.29	21 693.70 ± 38.14
$G_{[G_N\in\mathbb{R}]}$	21 639.74 ± 37.69	21 677.50 ± 39.46	21 699.69 ± 41.53

We can observe, that no penalty function is able to make the solutions feasible using penalty $G_{[1\in 01]}$ due to the limited information. Interestingly, the results for $G_{[N_t\in\mathbb{N}_0]}$ show that with the additional information of constraint violations per turbine, feasible solution can be generated. However, the adaptive function loses feasible solutions during the optimization process. This is often caused by a small penalty factor α, which was reduced by the adaptive penalty function after finding feasible solutions. At this stage of the optimization process, the step size is often relatively small and the EA would require numerous steps to move a turbine into the feasible area again. Unfortunately, as the optimization process only gets one Boolean value for each turbine from $G_{[N_t\in\mathbb{N}_0]}$, it cannot distinguish if the turbine is far away or near the feasible area and multiple steps to the feasible area are improbably. The constant and balanced penalty functions are able to keep the feasible solutions but there is no difference between both penalty functions. For the $G_{[G_N\in\mathbb{N}_0]}$ penalty, we can see that the enhanced information from G leads to a significant improvement applying the balanced function. For the adaptive function, in 3 of 100 runs, optimized feasible solutions were created, with a mean value of 21668.51.

Using a continuous co-domain, i.e., $G_{[N_t\in\mathbb{R}]}$ and $G_{[G_N\in\mathbb{R}]}$, all approaches are able to create optimized feasible solutions. Comparing the penalty functions, the balanced approach performs best. For $G_{[G_N\in\mathbb{R}]}$, comparing the results of the balanced and the adaptive penalty function leads to a p-value of $2.54 \cdot 10^{-4}$ using a Wilcoxon signed rank-sum test [6] which means the differences are significant. The p-value for the comparison of the constant and balanced penalty function is $3.45 \cdot 10^{-13}$. In case of the adaptive function with $G_{[N_t\in\mathbb{R}]}$ in 1 of 100 runs, and in case of $G_{[G_N\in\mathbb{R}]}$ in 3 of 100 runs, the optimized solutions have a penalty of more than 0.001, in both cases, this runs are not used for mean calculation.

TABLE V. Experimental results of Scenario 2 using random initial solutions.

	Penalty Function		
	Constant	Adaptive	Balanced
Penalty	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW
$G_{[1\in 01]}$	not feasible	not feasible	not feasible
$G_{[N_t\in\mathbb{N}_0]}$	22 668.58 ± 49.83	not feasible	22 658.88 ± 47.52
$G_{[G_N\in\mathbb{N}_0]}$	22 667.66 ± 48.87	not feasible	22 673.51 ± 42.38
$G_{[N_t\in\mathbb{R}]}$	22 667.82 ± 44.60	22 671.06 ± 48.05	22 687.39 ± 45.89
$G_{[G_N\in\mathbb{R}]}$	22 670.27 ± 50.42	22 665.49 ± 50.84	22 683.15 ± 43.57

The results from Scenario 2 shown in Table V confirm the observations from Scenario 1. But the differences are smaller than in Scenario 1. Thus, most single results are only significant when applying a p-value of 0.05 as significance threshold. But comparing all results for continuous penalties, the balanced penalty function significantly outperforms both other functions. Further, for the adaptive penalty function using penalty $G_{[G_N\in\mathbb{N}_0]}$ no feasible solutions are generated. Penalty functions that use continuous penalties are performing significantly better in Scenario 2. Again, for the adaptive penalty function with $G_{[G_N\in\mathbb{R}]}$, in 1 of 100 runs, the optimized solution has a penalty of more than 0.001.

B. Comparison with Feasible Initialization

In the next experiments, we analyze the behavior starting with a feasible initial solution. Table VI shows the results.

TABLE VI. Experimental results of Scenario 1 using feasible initial solutions.

	Penalty Function		
	Constant	Adaptive	Balanced
Penalty	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW
$G_{[1\in 01]}$	21 642.90 ± 41.25	not feasible	21 640.44 ± 36.36
$G_{[N_t\in\mathbb{N}_0]}$	21 637.61 ± 44.81	not feasible	21 638.59 ± 48.58
$G_{[G_N\in\mathbb{N}_0]}$	21 645.41 ± 40.18	most not feasible	21 690.34 ± 47.17
$G_{[N_t\in\mathbb{R}]}$	21 640.42 ± 42.39	21 663.51 ± 42.00	21 693.54 ± 36.21
$G_{[G_N\in\mathbb{R}]}$	21 646.28 ± 39.30	21 662.01 ± 49.91	21 690.30 ± 43.94

We can observe that the constant and the balanced penalty function behave similarly to the variants starting with random initial solutions, except when using $G_{[1\in01]}$. Here, death penalty is not able to find feasible solutions. The similar behavior is confirmed with p-values smaller than 0.001. We observe that the performance of the adaptive penalty function is slightly worse than the approach using random initial solutions with less significant p-values between 0.01 and 0.03. In case of the adaptive penalty function and $G_{G_N\in\mathbb{N}_0}$, in 5 of 100 runs, optimized feasible solutions were created, with a mean value of 21663.69. In case of $G_{[N_t\in\mathbb{R}]}$, in 4 of 100 runs, and for $G_{[G_N\in\mathbb{R}]}$ in 1 of 100 runs, the optimized solutions have a penalty of more than 0.001.

TABLE VII. EXPERIMENTAL RESULTS OF SCENARIO 2 USING FEASIBLE INITIAL SOLUTIONS.

	Penalty Function		
	Constant	Adaptive	Balanced
Penalty	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW	Mean ± Std $P \pm P$ in kW
$G_{[1\in01]}$	22 653.96 ± 49.11	not feasible	22 662.29 ± 52.41
$G_{[N_t\in\mathbb{N}_0]}$	22 654.02 ± 52.18	not feasible	22 656.16 ± 52.01
$G_{[G_N\in\mathbb{N}_0]}$	22 653.87 ± 57.39	not feasible	22 665.52 ± 47.71
$G_{[N_t\in\mathbb{R}]}$	22 654.82 ± 52.50	22 664.92 ± 51.43	22 677.67 ± 41.37
$G_{[G_N\in\mathbb{R}]}$	22 650.68 ± 50.46	22 663.90 ± 47.85	22 673.21 ± 51.06

Again, the results of Scenario 2 approve the observations from Scenario 1 qualitatively, but in this case, the differences are even less significant. Also here, for the adaptive penalty function using penalty $G_{[G_N\in\mathbb{N}_0]}$ no feasible solutions are generated and penalty functions with continuous penalties are performing significantly better. For the adaptive penalty function and $G_{[N_t\in\mathbb{R}]}$, in 1 of 100 runs, and for $G_{[G_N\in\mathbb{R}]}$ in 1 of 100 runs, the optimized solutions have a penalty of more than 0.001.

C. Analysis of Optimization Run

In this section, we analyze the behavior during the optimization runs. From the 100 runs for each configuration, we consider the runs with the median results. In the first step, we look at the behavior of the penalty factor α.

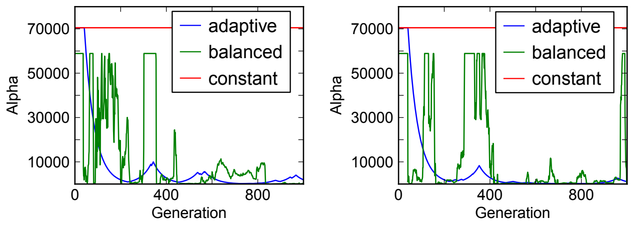


Fig. 6. Behavior of the penalty factor α with $G_{[N_t\in\mathbb{R}]}$ (left) and $G_{[G_N\in\mathbb{R}]}$ (right)

Figure 6 shows α for the constant, adaptive, and balanced penalty function. On the left side, $G_{[N_t\in\mathbb{R}]}$ is used and on the right side, $G_{[G_N\in\mathbb{R}]}$ is applied both with random initialization. As expected, the constant penalty function does not change α. We can see that the adaptive and balanced penalty function use a high value for α at the beginning of the optimization process and both decrease α after finding the first feasible solution. During the optimization, there are situations, where many infeasible solutions are in the population and the penalty functions increase α. The balanced penalty function adapts α very fast, while the adaptive penalty function changes α slower than the balanced.

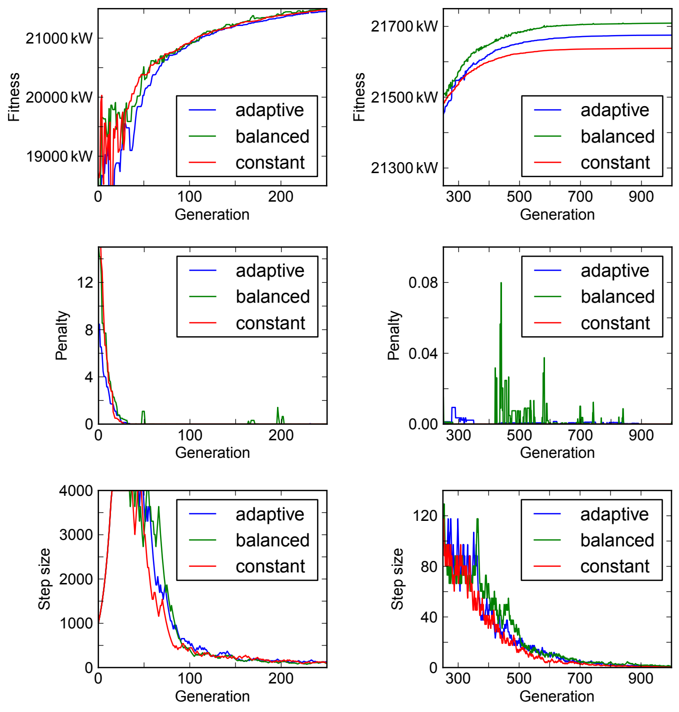


Fig. 7. Behavior of the fitness f, penalty G, and step size σ.

In the next figure, we focus only on the median runs of $G_{[G_N\in\mathbb{R}]}$. Figure 7 shows the behavior of the fitness f, penalty G, and step size σ. On the left side, we can see the first 250 generations, on the right side, the last 750 generations. The plotted function values are the results of the best solutions $\mathbf{x}^*$ with $\widetilde{f}(\mathbf{x}^*) \geq \widetilde{f}(\mathbf{x})$ for every $\mathbf{x}$ of the current population p. At the beginning, the fitness $f(\mathbf{x}^*)$ is varying strongly. After about 30 generations, the results become more stable, while after about 100 generations, the fluctuations decrease. This corresponds to the observation of penalty $G(\mathbf{x}^*)$, where after about 30 generations, the first feasible solutions are found. After less than 100 generations, only the adaptive and the balanced penalty functions use solutions $\mathbf{x}^*$ with $G(\mathbf{x}^*) > 0$. But the balanced approach employs clearly higher values for $G(\mathbf{x}^*)$. The step size σ is increased at the beginning up to its maximum value, which corresponds to the map size. After about 50 generations, it starts to decrease. We can observe that the step size in the balanced approach decreases slower in comparison to the other approaches.

D. Placement Results

The left part of Figure 8 shows the best optimized solution for Scenario 1 created by a balanced penalty function with penalty type $G_{[G_N\in\mathbb{R}]}$ and random initial solution. The symbols of the turbines visualize their positions, their heights do not reflect the true proportions. Most importantly, a feasible solution was created[2]. We can observe a clear organization of the turbine locations within the feasible area. To interpret the result, we need to keep in mind the wind distribution, which mainly blows from south-west, see detailed distribution shown in Figure 2. Three lines of turbines are turned towards the wind. The first line, w. r. t. to the main wind direction,

[2]The turbine T16 is placed on a small feasible island.

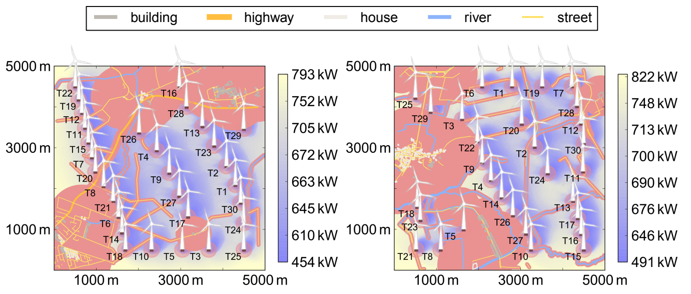

building
highway
house
river
street
5000 m
3000 m
1000 m
1000 m
3000 m
5000 m
793 kW
752 kW
705 kW
672 kW
663 kW
645 kW
610 kW
454 kW
822 kW
748 kW
713 kW
700 kW
690 kW
676 kW
646 kW
491 kW
T22 T19 T12 T11 T15 T7 T20 T8 T21 T6 T14 T18 T10 T5 T3 T25
T16 T28 T13 T29 T26 T4 T23 T9 T2 T1 T27 T30 T17 T24
T25 T29 T3 T6 T1 T19 T7 T28 T20 T12 T22 T2 T30 T9 T4 T24 T11
T14 T13 T18 T23 T26 T17 T5 T27 T16 T21 T8 T10 T15

Fig. 8. Left: best solution for Scenarios 1 with balanced penalty function using $G_{[G_N \in \mathbb{R}]}$ and random initial solution, right: best solution for Scenario 2 with balanced penalty function using $G_{[N_t \in \mathbb{R}]}$ and feasible initialization

employs the largest number of turbines. As the wind mainly comes from south-west, these turbines are very little affected by wake effects, but cause wake effects. For this reason, the distance between the first and the second line is larger than the distance between the second and the third line. Fewer turbines are placed in the second line than in the third one. The turbines in the third line face more towards the direction, where the wind comes from. Hence, turbines T16 and T28 can use places at the border of the constraints allowing T29 to be located behind the third line. Also the bottom of the scenario is used by the turbines T10, T5, and T3.

The right part of Figure 8 shows the best solution achieved for Scenario 2, which was generated with the balanced penalty function using $G_{[N_t \in \mathbb{R}]}$ and feasible initialization. The placement result shows similar properties like the previous one, in particular a feasible solution. Again, we can observe well-organized lines of turbines that exploit the effects of the main wind direction. In the south-west area, a feasible region is effectively used by a small number of turbines.

VII. Conclusion

In the turbine placement problem, geographical constraints induce a complicated optimization problem that require the application of advanced constraint handing methods. In this paper, we have shown that the employment of penalty functions that allow infeasible solutions during the optimization process significantly improves the results. We combined three types of penalty functions with five penalty types modeled for geographical constraints. Our experimental analysis revealed that the balanced penalty function performs best. It allows the immediate adaptation of penalty factor α to changes of the feasibility ratio of the population. The best solution for Scenario 1 has been generated with the balanced penalty function using $G_{[G_N \in \mathbb{R}]}$ and random initialization, the best solution for Scenario 2 also with the balanced variant, but with $G_{[N_t \in \mathbb{R}]}$ and feasible initialization. We did not observe significant differences between the two initializaiton variants, i.e., starting with random or with feasible solutions, except for the variants with $G_{[1 \in 01]}$.

As future work, we plan a comparison to further constraint handling methods, i.e., multi-objective approaches, repair functions, and meta-modeling of the contraint functions. Further, we plan to test a hybrid approach using genetic algorithm and differential evolution like proposed in [17].

Acknowledgment

We thank the German Weather Service for providing the COSMO-DE data and the Ministry for Science and Culture of Lower Saxony for supporting this work with the PhD program *System Integration of Renewable Energy (SEE)*. Also, this work has been supported by the ARC Discovery Early Career Researcher Award DE160100850.

References

[1] H. Beyer and H. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[2] C. A. Coello Coello. Constraint-handling using an evolutionary multi-objective optimization technique. *Civil Engineering and Environmental Systems*, 17:319–346, 2000.

[3] German Weather Service. COSMO-DE: numerical weather prediction model for Germany, 2012. http://tinyurl.com/dwd-cosmo-de.

[4] T. Höfer, Y. Sunak, H. Siddique, and R. Madlener. Wind farm siting using a spatial analytic hierarchy process approach: A case study of the städteregion aachen. *Applied Energy*, 163(C):222–243, 2016.

[5] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In *1st IEEE Conference on Evolutionary Computation*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[6] G. Kanji. *100 Statistical Tests*. SAGE Publications, 1993.

[7] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[8] O. Kramer. A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing*, 2010:3:1–3:19, 2010.

[9] O. Kramer, U. Schlachter, and V. Spreckels. An adaptive penalty function with meta-modeling for constrained problems. In *IEEE Congress on Evolutionary Computation, CEC*, pages 1350–1354, 2013.

[10] A. Kusiak and Z. Song. Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, 35(3):685–694, 2010.

[11] D. Lückehe, M. Wagner, and O. Kramer. On evolutionary approaches to wind turbine placement with geo-constraints. In *Genetic and Evolutionary Computation Conference, GECCO*, pages 1223–1230, 2015.

[12] R. Luebbe and B. Finch. Theory of constraints and linear programming: a comparison. *International Journal of Production Research*, 30(6):1471–1478, 1992.

[13] P. Neis, D. Zielstra, and A. Zipf. The street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011. *Future Internet*, 4(1):1–21, 2011.

[14] H. Neustadter. Method for evaluating wind turbine wake effects on wind farm performance. *Journal of Solar Energy Engineering*, pages 107–240, 1985.

[15] M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature, PPSN*, volume 1141 of *LNCS*, pages 245–254. Springer, 1996.

[16] The Wind Power. Wind farms in Lower Saxony, Germany, 2015. http://tinyurl.com/parks-lower-saxony.

[17] A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl. Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, 23:50–64, 2015.

[18] United Nations Climate Change Conference. COP21, 2015. http://www.cop21.gouv.fr/en/.

[19] M. Wagner, J. Day, and F. Neumann. A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy*, 51(0):64–70, 2013.

[20] M. Wagner, K. Veeramachaneni, F. Neumann, and U.-M. O'Reilly. Optimizing the layout of 1000 wind turbines. In *European Wind Energy Association Annual Event*, 2011.

[21] W. Weibull. A statistical distribution function of wide applicability. *Applied Mechanics, Transactions of ASME*, 3(18):293–297, 1951.