Negative Cycle Canceling with Neighborhood Heuristics for the Wind Farm Cabling Problem

Sascha Gritzbach
sascha.gritzbach@kit.edu
Karlsruhe Institute of Technology
Germany

Dorothea Wagner
dorothea.wagner@kit.edu
Karlsruhe Institute of Technology
Germany

Matthias Wolf
matthias.wolf@kit.edu
Karlsruhe Institute of Technology
Germany

ABSTRACT

The Wind Farm Cabling Problem (WCP) aims at finding the cost-minimal inter-array cable routing, also known as internal cable layout, of a wind farm so that all turbine generation is transmitted to the substations. For each possible connection in the wind farm, one of several cable types can be selected. Each cable type comes with a thermal capacity and unit length costs. WCP can be modeled as a graph theoretic minimum-cost flow problem with a step-cost function on each edge. We extend a deterministic "hill-climbing" heuristic from the literature. This heuristic runs into local minima from which it is not able to recover. We embed this algorithm into a framework which involves strategies for escaping these minima. These escaping strategies allow the heuristic to descend into other, possibly better, minima. We design three such strategies and provide an extensive statistical evaluation comparing these strategies. The best combination of strategies is evaluated against Gurobi 9.0.0 on a Mixed-integer Linear Program formulation and a Simulated Annealing-based heuristic from the literature on publicly available synthetic benchmark sets. Our simulations show that our framework works exceptionally well on the largest benchmark instances where it provides better solution within 15 minutes than Gurobi within one day on 80 % of the input instances. The simulations on the benchmark sets are complemented by a case study on the world's soon-to-be largest offshore wind farm: Hornsea One.

CCS CONCEPTS

• **Mathematics of computing → Network optimization**; *Graph algorithms*; *Network flows*.

KEYWORDS

Iterated Local Search, Negative Cycle Canceling, Step Cost Function, Wind Farm Planning

ACM Reference Format:
Sascha Gritzbach, Dorothea Wagner, and Matthias Wolf. 2020. Negative Cycle Canceling with Neighborhood Heuristics for the Wind Farm Cabling Problem. In *The Eleventh ACM International Conference on Future Energy Systems (e-Energy'20), June 22–26, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3396851.3397754

e-Energy'20, June 22–26, 2020, Virtual Event, Australia

ACM ISBN 978-1-4503-8009-6/20/06…$15.00
https://doi.org/10.1145/3396851.3397754

1 INTRODUCTION

Offshore wind energy, according to the International Energy Agency, has the potential to cover over 18 times today's worldwide electricity demand [12]. In October 2019, the last of 174 wind turbines of the Hornsea One wind farm was built. It is scheduled to be completed in 2020, at which point it will be the world's largest operational offshore wind farm [16, 19]. One step in planning an offshore wind farm is designing the internal cable layout, i. e., determining what cables should be used so that all turbine generation can be transmitted to the offshore substations. We refer to this planning step as the Wind Farm Cabling Problem (WCP). Since this problem is computationally difficult, a variety of exact and heuristic approaches exist to solve variants of WCP with different degrees of technical depth. This work extends a heuristic that used Negative Cycle Canceling (NCC) for a cost-minimization variant of WCP. We develop and evaluate strategies to escape local minima in which the heuristic gets stuck. Simulations on synthetic benchmark sets from the literature show that our extension strictly improves the original algorithm and that it yields better solutions than competitors but in a fraction of their allotted running time. In a case study we compare our algorithm to an exact approach using Mixed-integer Linear Programming (MILP) on the Hornsea One wind farm.

2 RELATED WORK

A wind farm can be modeled as a graph in which vertices correspond to turbines and substations, and edges represent possible connections between them. In an early work using such a graph model for WCP, the problem of finding a good cable layout is split into hierarchical layers. These layers relate to well-known graph problems for which heuristic approaches are proposed [3]. Other solution methods include clustering [6] or Simulated Annealing (SA) [13] or use similar problem formulations such as a Planar Open Vehicle Routing Problem formulation [1]. WCP can also be modeled as a graph-theoretic minimum-cost flow problem in which multiple cable types give rise to a step cost function on the edges [7]. Such step cost functions are the main difference to standard minimum-cost flow problems, in which flow incurs costs proportional to its magnitude. The authors of [7] present a hybrid approach of MILP and heuristic steps to fix constraint violations. They also provide an overview of other optimization problems in wind farm planning. We adapted Negative Cycle Canceling, a standard technique for minimum-cost flows with linear costs, for WCP [9, 10]. Combining a local search algorithm such as the NCC adaptation with strategies for escaping local minima is known as Iterated Local Search. An introduction to this notion can be found in [14]. This technique is applied to a Fixed-Charge Transportation Problem [4] and to a Water Distribution Network Design Problem [5].

3 CONTRIBUTION

In previous work, we showed that our NCC algorithm compares well against Mixed-integer Linear Programming and Simulated Annealing even though it gets stuck in local minima and is therefore not able to use its running time advantage [10]. In this work, we design strategies that allow the NCC algorithm to escape local minima and to continue its "hill-climbing" approach. We will denote the algorithm from [10] as the *standard NCC algorithm* or *NCC algorithm without escaping* and our extension as *NCC algorithm with escaping* or as *our framework*. In Section 4 we give a quick overview of the problem formulation. A recap of the standard NCC algorithm and a detailed explanation of our escaping strategies follow in Section 5, as well as a description how they combine with the standard NCC algorithm to constitute our framework. In Section 6 we present simulations on benchmark sets from [13], which also include a set of cable types. We identify the best mixture of our escaping strategies in Section 6.1, followed by a comparison to an MILP formulation solved by Gurobi (Section 6.2), to the standard NCC algorithm (Section 6.3), and to a heuristic using Simulated Annealing (Section 6.4). We conclude with a case study on a wind farm instance that resembles the aforementioned Hornsea One wind farm in Section 7. On small- and medium-sized instances, our framework proves to provide competitive solutions to other solution approaches On larger instances—instances the size of Hornsea One and bigger—it is able to find better solutions while simultaneously needing less running time.

4 WIND FARM CABLING PROBLEM

We describe the problem formulation according to [9]. The wind farm is modeled as a graph $G = (V, E)$, where the set of vertices V is partitioned into turbines V_T and substations V_S. Each turbine has a production of one unit and each substation has a capacity on the amount of turbine production units that can be collected by this substation. Between any two vertices, except for pairs of substations, there is at most one directed edge. The edge is directed arbitrarily. The edges represent possible connections on which a cable can be laid. A *flow* on G is a function $f\colon E \to \mathbb{R}$ with the following convention: If $f(u, v) > 0$ for some edge (u, v), we say that there is flow *from u to v*. For notational purposes we alias $f(v, u) = -f(u, v)$ for any edge (u, v), which explains how to interpret negative flow in a straight-forward way. We call a flow f on G feasible if all of the following conditions hold: 1) At every turbine, the difference of outgoing to incoming flow is exactly one (namely the production of the turbine). 2) At every substation, the total incoming flow is at most that substation's capacity. 3) No substation has an edge with flow leaving the substation.

The costs of a flow are derived from the available cable types. On each edge, the cost of the flow on that edge is the length of the edge times the per unit cost of the cheapest cable type that has sufficient capacity to carry the flow. The cost is zero if there is no flow on the edge and it is infinity if there is no cable type with sufficient capacity. This cost function is a step function and resembles the assumption that there can be at most one cable per edge. The *total cost* of a flow is the sum of the costs over all edges.

5 ALGORITHM

We describe the NCC algorithm as outlined in [9] and as explained in [10]. The algorithm first computes an initial feasible flow of finite cost. Then, the residual graph R is constructed as a copy of the wind farm graph G where each edge is replaced by two reversely oriented edges between the same vertices. A virtual substation is added that has one outgoing edge to and one incoming edge from every substation. The algorithm picks a natural number Δ and computes for each residual edge by how much the cost of the flow on the underlying wind farm edge would change if additional Δ units of flow were be sent in the direction of the residual edge. With the help of an adaptation of the well-known Bellman-Ford algorithm [2, 8], the NCC algorithm tries to find a negative cycle in the residual graph with the just mentioned marginal (or residual) costs. If it finds a negative cycle, the flow on G is adjusted according to the cycle and Δ. The algorithm continues with the potentially modified flow and a possibly new value of Δ until no more improvements can be found. In this case, the algorithm terminates in a local minimum. There are examples of non-optimal flows without negative cycles in the residual graph for any value of Δ.

The standard NCC algorithm terminates within two minutes on the benchmark sets from [13] and yields competitive results to an MILP solver on the same instances with a maximum running time of one hour [10]. In the conclusion of this work we raised the point that this gap in running times could be used to deal with the local minima the NCC algorithm gets stuck in. We pick up this suggestion and incorporate the NCC algorithm in a framework of strategies that change a flow and thereby potentially allow the NCC algorithm to descend into different local minima. In this section, we explain the framework and give detailed descriptions of these *escaping strategies*. For each strategy, a weight is specified which represents a strategy's frequency of being used in the course of our algorithm.

The framework starts by initializing a flow on the wind farm as specified by the NCC algorithm. This algorithm then performs its hill-climbing until no further negative cycle can be canceled. While the standard NCC algorithm terminates at this point, our framework picks one of its available escaping strategies randomly according to the specified weights and applies it to the current flow. If this strategy changes the flow (no matter if to the better or worse), another round of Negative Cycle Canceling is run until no more negative cycles are found. This round of negative cycle canceling is subject to adapted residual costs as explained in detail for each strategy. This adaptation helps to prevent changes made by escaping strategies from being overturned right away and is replaced by another adaptation once an escaping strategy successfully changes the flow. If an escaping strategy is not able to change the flow, another escaping strategy is picked and the unsuccessful strategy will not be picked again until the current flow has been changed. We refer to one pick of an escaping strategy with the ensuing NCC run (in case of changes) as one *iteration* of the framework. The framework terminates if no escaping strategy is able to change the current flow anymore or if a given time limit is exceeded.

In the following paragraphs we describe three escaping strategies we developed. All escaping strategies modify the flow based on local changes to small parts of the graph such as the neighborhood of a vertex. At all times will the flow stay feasible.

Free Upgrade (U). This strategy identifies all edges with a non-zero flow such that adding one additional unit of flow incurs the need for a bigger cable type. The strategy then performs a single run of the NCC algorithm with $\Delta = 1$ in which all residual costs are computed normally except for the aforementioned edges. Their residual costs are set to zero instead of a finite positive residual cost. If a cycle is canceled, all residual edges with a previously saturated cable type on that cycle are identified. For these edges, the residual cost for the remainder of the iteration are adjusted to reflect the free upgrade, i. e., the cost for the new cable type and all cable types with more capacity is reduced by the cost for the upgrade.

Move Leaf (L). This strategy identifies all leaves, i. e., turbines without incoming flow. It iterates over all leaves one-by-one and checks if it has a shorter outgoing edge than the one that transmits the turbine's production. In that case, one unit of flow is rerouted via the shorter edge to a substation with free capacity using only those edges whose flow is non-zero and less than the maximum cable capacity—if such a path exists. Then, for the remainder of this iteration, the residual costs of the leaf's new outgoing edge are adjusted in the sense that the cost for the cheapest cable type is given for free.

Deal with Bonbon (B). The adaptation of the Bellman-Ford algorithm used in the standard NCC algorithm may also identify negative sets of cycles in the residual graph which do not improve the flow when being canceled [10]. The reason for those sets is that negative cycles consisting of two edges only may exist. Even if the negativity suggest a decrease of the total costs, canceling such a two-edge cycle does not change the flow at all. A drawing of such a set of cycles in [9] inspired us to call such a set of cycles a *bonbon*. This escaping strategy makes use of such an unhelpful set of cycles as follows: It first identifies all negative residual edges on the bonbon. Then it tries to find paths in the residual graph that close a negative cycle when merged with a negative edge (and possibly other edges) from that bonbon. The strategy therefore runs again the Bellman-Ford algorithm with the same setting (including the adapted residual costs if applicable) for which the unhelpful bonbon occurred. For each negative edge on the bonbon it considers each incoming edge.From there it traverses the parent pointers out of the Bellman-Ford algorithm until a cycle is closed. If the cycle is negative, then it is canceled and the escaping strategy terminates. Otherwise, the search continues. This strategy does not change the standard residual cost computation for the following NCC run.

6 SIMULATIONS

The framework is written in C++14 and compiled with GCC 8.2.1 using the `-O3 -march=native` flags. All simulations run in single-thread mode (to ensure comparability) on a 64-bit architecture with four 12-core AMD-CPUs clocked at 2.1 GHz with 256 GB RAM running OpenSUSE Leap 15.1.

We evaluate our framework using the cable types from Table 1 on publicly available benchmark sets [13]. The instances from benchmark set $\mathcal{N}_1$ have one substation and 10–79 turbines. The other benchmark sets contain instances with multiple substations and varying numbers of turbines: 20–79 for $\mathcal{N}_2$, 80–180 for $\mathcal{N}_3$ and $\mathcal{N}_5$, and 200–499 for $\mathcal{N}_4$. The instances in $\mathcal{N}_5$ are complete graphs except that edges between substations are omitted. The edge sets of instances in all other benchmark sets are generated by a nearest-neighbor procedure with additional shortcuts. The parameters for this procedure are the same in all those benchmark sets.

Table 1: The cable types from [13].

Cable type	1	2	3	4
Capacity	5	8	12	15
Cost per unit length	20	25	27	41

Table 2: Number of instances where escaping strategies yield better solutions than the NCC algorithm without escaping.

B	L	LB	U	UB	UL	ULB
87	434	460	297	327	580	603

Table 3: Comparison of sets of escaping strategies. An entry in row i and column j shows on how many instances setting i produces better solutions than setting j. Values are marked by a star if they are significant with $p < 10^{-2}$ and by two stars if $p < 10^{-4}$. The best set of strategies is marked green.

	B	L	LB	U	UB	UL	ULB
B	—	8.9 %	0 %	17.8 %	8.5 %	4.3 %	0.3 %
L	91.9 %**	—	23.3 %	70.6 %**	65.7 %**	14.9 %	14.2 %
LB	100 %**	76.7 %**	—	74.8 %**	71.9 %**	23.0 %	16.9 %
U	82.2 %**	29.4 %	25.2 %	—	1.8 %	1.0 %	2.7 %
UB	91.5 %**	34.3 %	28.1 %	98.2 %**	—	6.5 %	4.0 %
UL	95.7 %**	85.1 %**	77.0 %**	99.0 %**	93.5 %**	—	36.1 %
ULB	99.7 %**	85.8 %**	83.1 %**	97.3 %**	96.0 %**	63.1 %*	—

6.1 Comparing sets of escaping strategies

In a first step, we want to compare different sets of escaping strategies to see how they perform in helping the NCC algorithm to move away from local minima. We consider seven different sets. The sets are given by the three escaping strategies from Section 5 with each escaping strategy having either weight 0 or 1. The all-zero set is not considered. We refer to the sets by the letters from each strategy in the set with weight 1. Hence, the set referred to as UB uses the strategies *Free Upgrade* and *Deal with Bonbon* with a weight of 1 each but does not use the strategy *Move Leaf*. For the simulations, we randomly select 200 instances from each of the five benchmark sets in [13]. Each set of strategies is run on each instance with a time limit of 15 minutes and the best solution value for the combination of strategy set and instance is recorded. In the terminology of [10] we use `CollectingDijkstraAny` as the initialization and `IncDec` as the delta strategy. Those have been identified as the best strategies for the standard NCC algorithm [10]. Table 2 shows the numbers of instances out of the selected 1000 on which each of the sets of escaping strategies yields a better solution to WCP than the standard NCC algorithm. For the *Deal with Bonbons* strategy this count equals the number of instances where there was a change to the flow at all. The other two strategies rely on the respective ensuing run of the NCC algorithm to find better solutions. In particular, for those strategies, a change to the flow does not necessarily yield an improvement to the best solution. The numbers in Table 2 show that *Deal with Bonbons* and *Free Upgrade* seem to have the most

problems to allow finding better solutions. *Move Leaf* on its own allows finding better solutions on more than 43 % of the instances. But only in conjunction with the other escaping strategies, better solutions can be found on approximately 60 % of all instances.

These numbers only show the presence of a new better solution after applying escaping strategies. We need to investigate how the new best solutions of a set compare to solutions from the other sets.

For each ordered pair (i, j) of set of strategies we count the number of instances n_i and n_j where the best solution found by i (and j, respectively) is better than the one found by j (and i, respectively). If i and j were equally good, then $n_i \sim \mathrm{Bin}(n_i + n_j, \theta)$ with $\theta = 0.5$ for each set of randomly and independently chosen instances, i. e., n_i is binomially distributed. For each ordered pair we perform a one-sided binomial sign test for two dependent samples [18]. We test the null hypothesis $\mathrm{H}_0\colon \theta = 0.5$ against the alternative hypothesis $\mathrm{H}_1\colon \theta > 0.5$. We apply a Bonferroni-correction by the number of tests (42). We interpret rejecting the null hypothesis as setting i performing better than setting j. In Table 3 we show the ratios n_i/n_i+n_j and the corresponding significance levels. Note that symmetric entries need not represent all 1000 instances since instances are omitted if both strategies find the same best solution.

Interestingly, the set of strategies which looked most promising in quantity according to Table 2 also turns out to provide better solutions than all the other sets. The difference between ULB and UL is quite small: Only on 295 out of 1000 instances there is a different best solution. Nonetheless, the results from Tables 2 and 3 show that ULB is the best set of strategies among those we considered in our simulations. We therefore use this set of escaping strategies when we compare our framework to other approaches to WCP.

6.2 Comparing NCC with Escaping to MILP

Next, we want to see how the framework compares to an approach using Mixed-integer Linear Programming. To this end, we randomly select 200 instances from each of the five benchmark sets independently from the previous selection. We run our framework with setting ULB on each instance five times with different random seeds and a time limit of 15 minutes each. For the MILP experiments we use Gurobi 9.0.0 [11] on the following formulation with a maximum running time of one day[1]:

$$\min \sum_{e\in E}\sum_{k\in K} c_k \cdot x(e,k)\cdot \mathrm{len}(e) \tag{1}$$

$$\text{s. t. } f_{\mathrm{net}}(u) = -1 \qquad \forall u \in V_T, \tag{2}$$

$$f_{\mathrm{net}}(v) \le \mathrm{cap}_{\mathrm{sub}}(v) \qquad \forall v \in V_S, \tag{3}$$

$$|f(e)| \le \sum_{k\in K} x(e,k)\cdot \mathrm{cap}_k \forall e \in E, \tag{4}$$

$$\sum_{k\in K} x(e,k) \le 1 \qquad \forall e \in E, \tag{5}$$

$$f(u,v) \le 0 \qquad \forall (u,v)\in E\colon u \in V_S, \tag{6}$$

$$f(u,v) \ge 0 \qquad \forall (u,v)\in E\colon v \in V_S, \tag{7}$$

$$f(e) \in \mathbb{R} \qquad \forall e \in E, \tag{8}$$

$$x(e,k) \in \{0,1\} \qquad \forall e \in E, k \in K, \tag{9}$$

[1]The MILP approach is an exact solution method. Yet, it is computationally prohibitive on large instances to run Gurobi until the optimal solution is provably found.

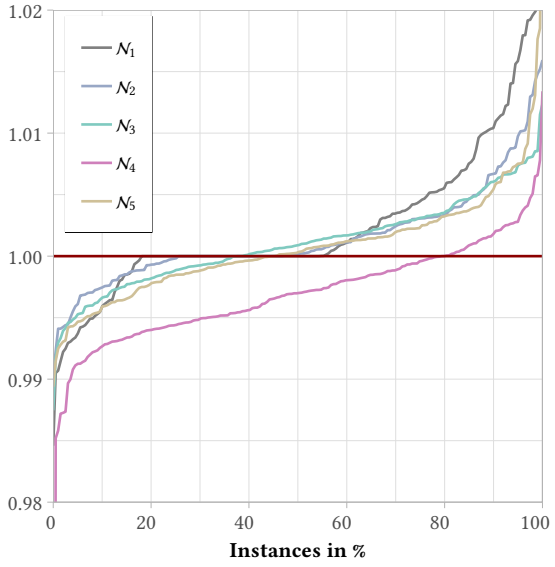


Figure 1: Ratio of solution values from our framework and Gurobi separated by benchmark sets. On each input instance our framework runs five times with different random seeds and the best solution from each run is recorded. The value of worst of these five solution from each run is divided by the best solution found by Gurobi. These ratios are depicted in increasing order.

where $G = (V_T \cup V_S, E)$ is the wind farm graph and $\mathrm{cap}_{\mathrm{sub}}(v)$ is the capacity of substation v. The length of an edge e is denoted by $\mathrm{len}(e)$ and the net flow at a vertex u is given by $f_{\mathrm{net}}(u) = \sum_{(v,u)\in E} f(v,u) - \sum_{(u,v)\in E} f(u,v)$. The set of available cable types is denoted by K. For each cable type $k \in K$ let c_k be the cost of k per unit of length and cap_k be its capacity. We interpret the decision variables as $x(e,k) = 1$ if and only if cable type k is used on edge e. Equation (5) ensures that at most one cable is built per edge. Equations (2) and (3) represent the balance constraints at turbines and substations, respectively, and that no substation capacity is exceeded. The cable capacity constraints are given by Eq. (4). Finally, Eqs. (6) and (7) stand for the assumption that no flow leaves substations back into the wind farm.

In Figure 1 we show a comparison of solution values from our framework and the MILP. For each instance, we identify the worst of the five separate best solutions after 15 minutes and use this value for the comparison to Gurobi. The reason for this choice is to account for different trajectories of picks of escaping strategies. Picking the worst of the five solutions relates to a worst-case analysis of our framework. We compute the ratio of this "worst-case value" to the best solution found by Gurobi at the conclusion of its one-day running time. Hence, a value less than 1 means that in all five runs of our framework the best solution found is better than the solution given by Gurobi. We order these ratios in increasing order and plot them separately for each benchmark set. Two values are not shown: In benchmark set $\mathcal{N}_4$ there is one data point at 0.9797 and in benchmark set $\mathcal{N}_1$ there is one data point at 1.0303.

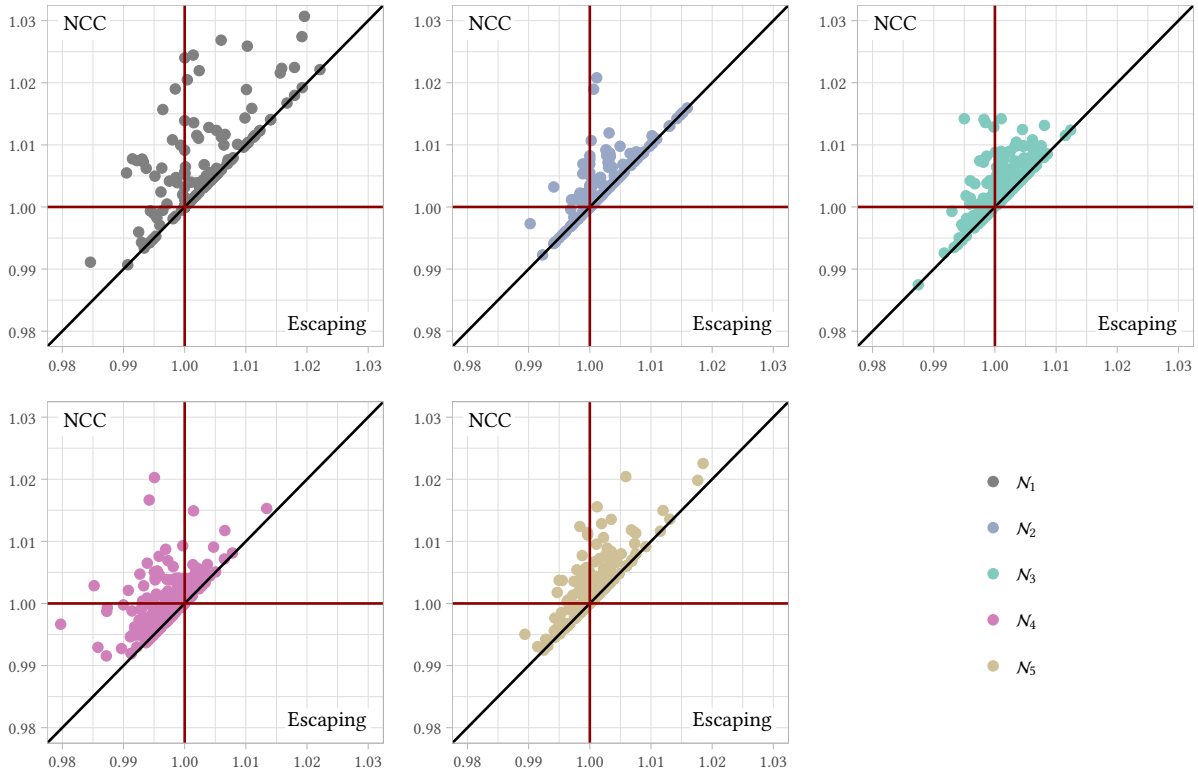


Figure 2: Ratios of best solutions computed by NCC algorithms to best solutions found by Gurobi separated by benchmark sets. Ratios on the x-axis show solutions from the NCC algorithm with escaping strategies and ratios on the y-axis show solutions from the standard NCC algorithm, i. e., without escaping strategies. The NCC algorithm with escaping is run five times with different random seeds and only the worst run is considered.

In the two smallest benchmark sets $\mathcal{N}_1$ and $\mathcal{N}_2$ our framework and Gurobi find the same solution on a combined 27.75 % of the randomly selected instances and our framework finds better solutions on 22 % of the instances. On benchmark sets $\mathcal{N}_3$ and $\mathcal{N}_5$, whose graphs only differ in the number of edges, the NCC algorithm with escaping finds better solutions on 37.5 % and 45 % of the instances. It performs best on the benchmark set $\mathcal{N}_4$, which includes the instances with the highest number of turbines (up to 500). Here, it outperforms Gurobi on 79.5 % of the selected instances and achieves the best solution ratios across all benchmark sets.

6.3 Improvement over standard NCC

From the construction of our framework it is obvious that it does not perform worse than the standard NCC algorithm. We want to see by how much our framework improves the solutions from the algorithm without escaping. To this end, we run the standard NCC algorithm on the same 1000 instances we have used for the MILP experiments. As stated in [10], the standard NCC algorithm terminates in less than two minutes. A comparison of these solution values to solution values from our framework after a running time of 15 minutes is shown in Figure 2. Each data point corresponds to one instance. On the x-axis we display the solution ratios from Figure 1. On the y-axis the ratios of the standard NCC algorithm to the MILP are depicted. As before, values below 1 stand for instances on which the respective NCC algorithm performs better than the MILP. Values above the diagonal represent instances on which the escaping strategies yields better solutions than NCC without escaping. Figure 2 shows that on 36.5 % and 32.5 % of the instances from $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively, the NCC algorithm with escaping provides better solutions than the standard algorithm. On $\mathcal{N}_3$ and $\mathcal{N}_5$ these figures rise to 69 % and 71.5 %, respectively. On the biggest instances ($\mathcal{N}_4$), our framework improves the standard NCC solution on 85 % of the selected instances. A total of 25 data points are not depicted in Figure 2, all of which are due to the fact that the standard NCC solution yields a ratio bigger than 1.03. In all those instances the NCC variant with escaping provides strictly better solutions than the standard variant. Six of those data points are from $\mathcal{N}_1$, with NCC ratios between 1.0307 and 1.0433. The corresponding ratios from our framework are between 1 and 1.0304. A single data point is from $\mathcal{N}_4$: (0.9923, 1.0451), which means that the NCC algorithm with escaping finds better solutions than the MILP in all five runs. The remaining 18 data points are from $\mathcal{N}_5$. All standard NCC ratios but one are between 1.0644 and 1.1707 with corresponding ratios for the escaping algorithm between 0.9944 and 1.0362 (six of which are smaller than 1). The single remaining point has a NCC ratio of 1.9937, which reduces to 1.0014 after escaping.

6.4 Comparing NCC with Escaping to SA

We want to compare our framework to a different heuristic approach, namely an algorithm using Simulated Annealing [13]. To this end, we again select 200 instances from each of the five benchmark sets from [13]. As before, we perform five runs of our framework per instance for 15 minutes each with different random seeds. We also run the Simulated Annealing approach on each instance with a maximum running of one hour in its best setting [13].

In Figure 3 we show the results from those simulations. As before, we record the best solution from each of the five runs of our framework and identify the worst of those five solutions. We divide this solution value by the best solution found by the Simulated Annealing approach. The resulting ratios are ordered increasingly and displayed on the y-axis. Note that values below 1 stand for instances on which the NCC algorithm with escaping performed better than Simulated Annealing. It stands out that the benchmark sets separate in two categories: those with few edges ($\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$) and those with many edges ($\mathcal{N}_4$ and $\mathcal{N}_5$). On the smaller instances, Simulated Annealing performs better than NCC with escaping: On $\mathcal{N}_1$, Simulated Annealing finds better solutions on 47.5 % of the instances. On $\mathcal{N}_2$ and $\mathcal{N}_3$ this holds on 63 % and 62 %, respectively. There are a total of 122 out of 600 instances in those benchmark sets on which both approaches yield equal solutions. Even though Simulated Annealing performs better on instances from the three benchmark sets with smaller instances, in all but four instances our framework yields solutions that are at most 2 % worse than the solutions provided by Simulated Annealing—keeping in mind that we use the worst of five best solutions computed by our framework. For those benchmark sets, five data points are not depicted in Figure 3, all of which are from $\mathcal{N}_1$. The respective ratios range between 0.8984 and 0.9689. Looking at the benchmark sets whose instances have many edges, we see that the NCC algorithm with escaping strategies outperforms Simulated Annealing. It provides better solutions on 95.5 % ($\mathcal{N}_4$) and 93 % ($\mathcal{N}_5$) of the selected instances. On more than half of these instances, our algorithm is better than Simulated Annealing by a margin of 1 %. On 17.5 % this margin is bigger than 3 % (not depicted in Fig. 3).

We conclude that the answer to the question whether NCC with escaping or Simulated Annealing performs better heavily depends on the size of the input instance. Note, however, that in our simulations Simulated Annealing is allowed four times the maximum running time of the NCC algorithm and that we use the worst of five NCC algorithm runs for the comparison.

7 CASE STUDY: HORNSEA ONE

In the previous simulations we have seen that using the NCC algorithm with escaping strategies yields very good solutions compared to two other approaches from the literature. We want to see how our framework works on real-world data. To this goal we do a case study on the largest, soon operational, offshore wind farm in the world: Hornsea One off the coast of Great Britain [19].

We extract the geographical coordinates of the wind turbines and substations from Open Street Map [17]. From there, we create a complete graph (except for edges between pairs of substations as stated in Section 4). This results in a graph with 174 turbines, three substations and a total of 15573 edges. In reference to the benchmark

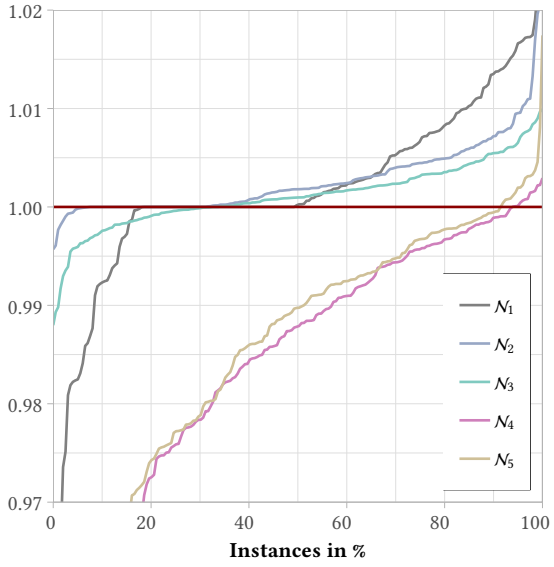


Figure 3: Ratio of solution values from our framework and Simulated Annealing separated by benchmark sets. On each input instance our framework runs five times with different random seeds and the best solution from each run is recorded. The value of worst of these five solution from each run is divided by the best solution found by the Simulated Annealing approach. These ratios are depicted in increasing order.

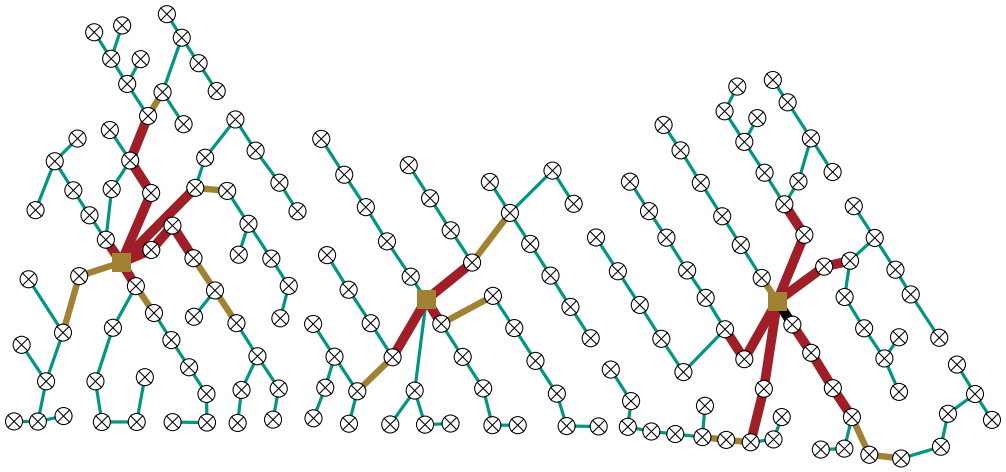
Table 4: Solution ratios of our framework to Gurobi on Hornsea One

Run 1	Run 2	Run 3	Run 4	Run 5
0.9966	0.9909	1.0015	0.9967	1.0013

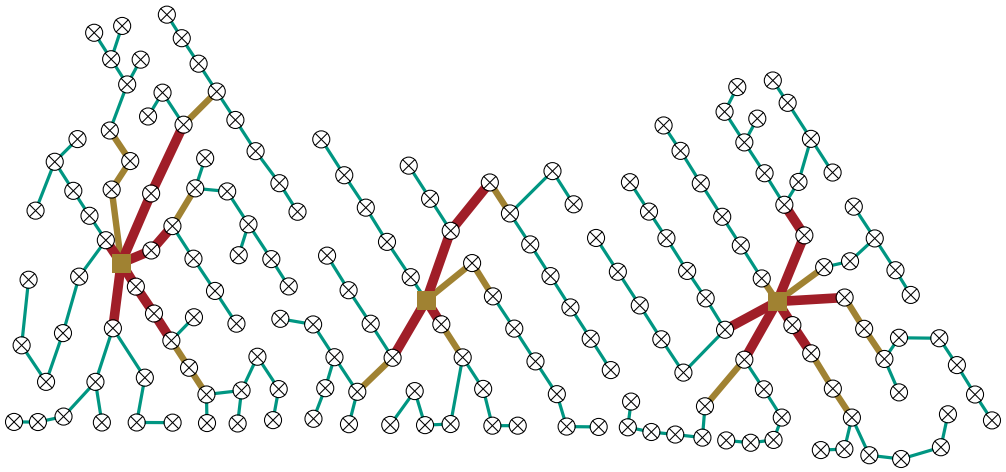
sets from [13], such an instance would be one the largest instances from $\mathcal{N}_5$. We use the cable types from these benchmark sets as specified in Table 1. For simplicity we use Euclidean distances for the edges instead of orthodromic distances.

As before, we give Gurobi one day of maximum running time. Our framework runs for 15 minutes each in five runs with different random seeds. In Table 4 we show the ratios of solution values from each of the five runs to the MILP solution. As before, ratios below 1 indicate that our framework finds a better solution than Gurobi. In three of five runs does our framework find a better solution than Gurobi. In the worst of the five runs, the solution given by the NCC algorithm with escaping is less than 0.2 % worse than the solution from the MILP. The best solution across five runs is nearly 1 % better than the solution computed by Gurobi.

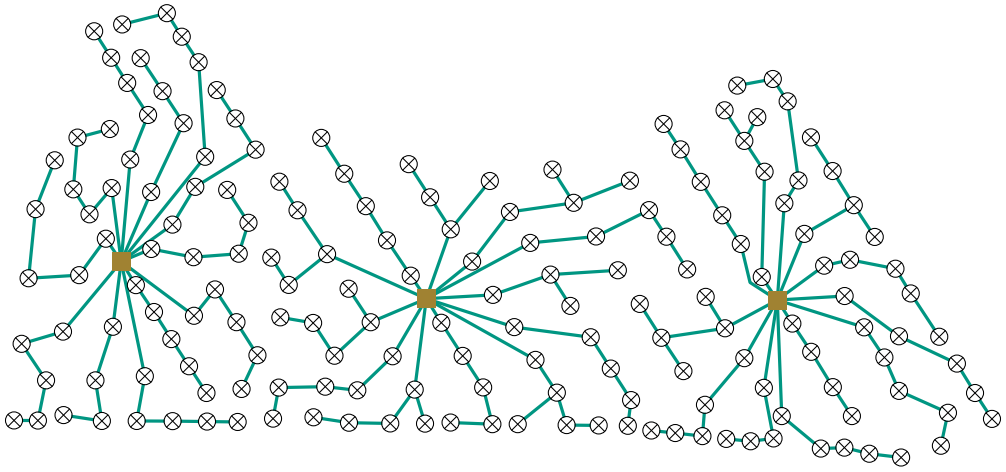
We want to visually compare different cablings: One from our framework, one from Gurobi and the real-world internal cabling as shown in [15]. For our algorithm we choose the cabling from run 3 as it performs worst of all runs. From top to bottom in Figure 4 we show the cabling computed by the NCC algorithm, by Gurobi and



(a) Cabling computed by our framework in Run 3



(b) Cabling computed by Gurobi



(c) Real-world cabling adapted to the cable types from the benchmark sets

Figure 4: Visualization of three internal cablings for the Hornsea One wind farm

the real-world cabling. We color-code the four cables types according to increasing capacity (green, orange, red, and black). For the real-world cabling only one cable type seems to be used. This one, however, fits very well in the cable types from the benchmark sets, as the smallest cable type has a capacity of five units of turbine production. In the real-world cabling, the maximum amount of turbine production on a single cable is also five. Note that the model as summarized in Section 4 does neither prohibit cycles nor cable crossings.

It stands out that both algorithmic approaches yield cablings with large proportions of radial layouts. Turbine production tends to be collected from the radial layout bits to make use of cable types with higher capacity the closer the nearest substation gets. From looking at different solutions it seems hard to identify parts of a solution which are particularly good compared to other solutions. There seem to be, however, some parts in particular near the boundary of the wind farm, where multiple solutions coincide. It could prove helpful to the algorithmic approaches to fix those parts in the solutions to some extent to facilitate finding the best solutions possible in the given running time.

8 CONCLUSION

We investigated if and to what extent a Negative Cycle Canceling-based hill-climbing algorithm [9, 10] for the Wind Farm Cabling Problem can profit from various strategies that allow the algorithm to escape local minima. To this end we developed three such strategies: *Free Upgrade* temporarily allows thicker cables without having to pay for the upgrade, *Move Leaf* allows a local change of a turbine without incoming flow, and *Deal with Bonbon* overcomes algorithmic difficulties from the negative cycle detection algorithm by altering structures on which flow is changed. Mixing these strategies yielded different degrees of helpfulness in escaping local minima. An equal mixture of all three proved most helpful among the settings we considered. We compared the NCC algorithm using this most helpful setting to Gurobi on an MILP formulation and to the standard NCC algorithm without escaping strategies. The escaping strategies do indeed improve the standard NCC algorithm. Not only do they yield solutions within 1 % of the solutions found by Gurobi after a running time of one day most of the time, but also provide better solutions on roughly 80 % of the largest instances we considered. In comparison to an approach using Simulated Annealing our framework did exceptionally well on larger instances or instances with many edges. With the results on the standard NCC algorithm in mind [10] the choice which of both NCC approaches to use strongly depends on the circumstances, e. g., on the available running time. If provably optimal solutions are desired and high running times are not prohibitive to the use case, then the MILP approach might be the go-to solution method. In a case study on the Hornsea One wind farm we compared solutions provided by our framework and by Gurobi. We have seen that they provide similar solutions in terms of solution costs but also that they are difficult to compare visually.

Moving forward, one could try to develop more escaping strategies to further facilitate exploring the search space and escaping local minima in the NCC algorithm to increase the numbers from Table 2. More tuning in the weights for the escaping strategies could further improve the solution qualities provided by our framework. From a more practical perspective, more characteristics than only the solution value could prove helpful in comparing different solutions given by various algorithmic approaches. Such characteristics could represent technical constraints such as non-crossing or cycle-free cable layouts. It remains open if and to what extent these constraints can be incorporated into the NCC algorithm. Along the same lines could it prove helpful for practitioners to incorporate other cost elements, e. g., losses or maintenance, into the cost function of the minimum-cost flow problem.

ACKNOWLEDGMENTS

This work was funded by the Helmholtz Association Program Storage and Cross-linked Infrastructures, Topic 6 Superconductivity, Networks and System Integration, by the Helmholtz Association future topic Energy Systems Integration, and by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation.

REFERENCES

[1] Joanna Bauer and Jens Lysgaard. 2015. The offshore wind farm array cable layout problem: a planar open vehicle routing problem. *Journal of the Operational Research Society* 66 (2015), 360–368. https://doi.org/10.1057/jors.2013.188

[2] Richard Bellman. 1958. On a Routing Problem. *Quart. Appl. Math.* 16 (1958), 87–90. https://doi.org/10.1090/qam/102435

[3] Constantin Berzan, Kalyan Veeramachaneni, James McDermott, and Una-May O'Reilly. 2011. *Algorithms for cable network design on large-scale wind farms.* Technical Report. Massachusetts Institute of Technology.

[4] Erika Buson, Roberto Roberti, and Paolo Toth. 2014. A Reduced-Cost Iterated Local Search Heuristic for the Fixed-Charge Transportation Problem. *Operations Research* 62, 5 (2014), 1095–1106. https://doi.org/10.1287/opre.2014.1288

[5] Annelies De Corte and Kenneth Sörensen. [n.d.]. An iterated local search algorithm for water distribution network design optimization. *Networks* 67, 3 ([n. d.]), 187–198. https://doi.org/10.1002/net.21673 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21673

[6] S. Dutta and T. J. Overbye. 2011. A clustering based wind farm collector system cable layout design. In *Power and Energy Conference at Illinois (PECI), IEEE.* 1–6. https://doi.org/10.1109/PECI.2011.5740480

[7] Martina Fischetti and David Pisinger. 2019. Mathematical optimization and algorithms for offshore wind farm design: an overview. *Business & Information Systems Engineering* 61, 4 (2019), 469–485.

[8] Lester R. Ford, Jr. and Delbert R. Fulkerson. 2010. *Flows in Networks.* Princeton University Press, Princeton, NJ, USA.

[9] Sascha Gritzbach, Torsten Ueckerdt, Dorothea Wagner, Franziska Wegner, and Matthias Wolf. 2018. Towards negative cycle canceling in wind farm cable layout optimization. In *Proceedings of the 7th DACH+ Conference on Energy Informatics*, Vol. 1 (Suppl 1). Springer. https://doi.org/10.1186/s42162-018-0030-6

[10] Sascha Gritzbach, Torsten Ueckerdt, Dorothea Wagner, Franziska Wegner, and Matthias Wolf. 2019. Engineering Negative Cycle Canceling for Wind Farm Cabling. In *27th Annual European Symposium on Algorithms (ESA 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Michael A. Bender, Ola Svensson, and Grzegorz Herman (Eds.), Vol. 144. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 55:1–55:16. https://doi.org/10.4230/LIPIcs.ESA.2019.55

[11] Gurobi Optimization, Inc. 2018. *Gurobi optimizer reference manual.* Gurobi Optimization, Inc. http://www.gurobi.com, Accessed: 2018-05-31.

[12] IEA. 2019. *Offshore Wind Outlook 2019.* IEA, Paris. Retrieved 2020-02-10 from https://www.iea.org/reports/offshore-wind-outlook-2019.

[13] Sebastian Lehmann, Ignaz Rutter, Dorothea Wagner, and Franziska Wegner. 2017. A Simulated-Annealing-Based Approach for Wind Farm Cabling. In *Proceedings of the Eighth International Conference on Future Energy Systems* (Shatin, Hong Kong) *(e-Energy '17).* ACM, New York, NY, USA, 203–215. https://doi.org/10.1145/3077839.3077843

[14] Helena Ramalhinho Lourenço, Olivier C. Martin, and Thomas Stützle. 2019. *Iterated Local Search: Framework and Applications.* Springer International Publishing, Cham, 129–168. https://doi.org/10.1007/978-3-319-91086-4_5

[15] 4C Offshore Ltd. 2020. *Global Offshore Renewable Map.* 4C Offshore Ltd. Retrieved 2020-02-07 from https://www.4coffshore.com/offshorewind/index.aspx?lat=53.883&lon=1.922&wfid=UK81

[16] Ørsted Hornsea Project One. 2019. *Final turbine installed as world's largest offshore wind farm nears completion.* Retrieved 2020-02-10 from https://hornseaprojectone.co.uk/News/2019/10/Final-turbine-installed-as-worlds-largest-offshore-wind-farm-nears-completion

[17] OpenStreetMap contributors. 2020. Planet dump. Retrieved 2020-02-08 from https://planet.openstreetmap.org

[18] David J. Sheskin. 2011. *Handbook of parametric and nonparametric statistical procedures* (5. ed. ed.). CRC Press, Taylor & Francis, Boca Raton [u.a.].

[19] Hanna Ziady. 2019. *The world's largest offshore wind farm is nearly complete. It can power 1 million homes.* Retrieved 2020-02-10 from https://edition.cnn.com/2019/09/25/business/worlds-largest-wind-farm/index.html