The Offshore Wind Farm Array Cable Layout Problem – A Planar Open Vehicle Routing Problem

J. Bauer, J. Lysgaard

Abstract

In an offshore wind farm, the turbines are connected to a transformer by cable routes which cannot cross each other. Finding the minimum cost array cable layout thus amounts to a vehicle routing problem with the additional constraints that the routes must be embedded in the plane. For this problem, both exact and heuristic methods are of interest. We optimize cable layouts for real-world offshore wind farms by a hop-indexed integer programming formulation, and develop a heuristic for computing layouts based on the Clarke and Wright savings heuristic for vehicle routing. Our heuristic computes layouts on average only 2% more expensive than the optimal layout. Finally, we present two problem extensions arising from real-world offshore wind farm cable layouts, and adapt the integer programming formulation to one of them. The thus obtained optimal layouts are up to 13% cheaper than the actually installed layouts.

Keywords: Offshore wind farm array cable layout, Planar open vehicle routing, Savings heuristic, Integer programming

Introduction

Offshore wind farms (OWFs) are rapidly gaining importance as power stations. The European Wind Energy Association predicts installed capacity in Europe to rise from currently about 4 GW to 150 GW by 2030 (Arapogianni et al., 2011). In order to harvest the generated power, all turbines are connected by array cable to an (on- or offshore) electrical transformer. Subsea cable, and its installation and maintenance, is very expensive. Thus, the cost of power produced by an OWF can be significantly reduced by minimizing the cost of the cable layout installed in an OWF.

For this problem, both exact and heuristic methods are of interest. For deciding the positions of the turbines in an OWF, many positions need to be computationally evaluated. For example, TopFarm, probably the most sophisticated tool to date for optimizing the positions of turbines in an OWF, evaluates more than 1000 turbine position configurations (Réthoré et al., 2011). The necessary quick estimation of the cable cost is best done by heuristic methods. Once the turbine positions are determined, an exact method finds the guaranteed cheapest cable layout, which then is installed.

The OWF Array Cable Layout (OWFACL) problem was introduced to us by a cable installing contractor as follows: Given turbine and transformer positions and a cable capacity in number of turbines, find a set of cable routes minimizing the total cable cost, connecting every turbine to a transformer, not exceeding cable capacity, and such that *cables do not cross each other.*

While it in theory might be possible to let cables cross, the problem was presented to us including the constraint not to let cables cross for two reasons: First, high voltage power cables generate heat, implying that two crossing cables would have to be insulated against each other. Second, the cables are buried into the seabed for their protection. For a cable crossing, one of two crossing cables would have to be buried below the other at the crossing. If the cable that is buried lowest fails and has to be replaced, both cables would have to be dug up, resulting in considerably higher costs.

Similarly, while it in theory is possible to let cables branch (resulting in a layout with a tree structure), the problem was presented to us as requiring a routing layout, since implementing cable branches necessitates non-standard parts, which are considerably more expensive.

This problem amounts to the well-known Open Vehicle Routing Problem (OVRP) (Li et al., 2007) with unit demands and additional *planarity constraints*. OVRP is NP-hard (Letchford et al., 2007). The OWFACL problem can be formulated as Planar OVRP (POVRP) as follows: Find a set of open vehicle routes minimizing the total route costs, connecting every client to a depot, not exceeding vehicle capacity, and such that the routes do not cross each other.

The electrical design of OWFs has to the best of our knowledge so far only been studied from an electrical engineering perspective using considerably varying problem definitions. Lumbreras and Ramos (2012) review this literature. Reviewed articles study not only the array layout, but also decisions on, among others, turbine types, transmission currents, and current transformations. Most reviewed articles artificially restrict the array layout, and use metaheuristics to solve problem instances.

To the best of our knowledge, vehicle routing problems with planarity constraints have not yet been studied.

According to the literature reviews by Subramanian (2012) and Li et al. (2007), most work on OVRP has focused on metaheuristics, with tabu search, simulated annealing, and ant colony being the most popular approaches. Apart from these, Sariklis and Powell (2000) propose a cluster first, route second heuristic.

Letchford et al. (2007) and Pessoa et al. (2008) use branch-and-cut to solve OVRP instances exactly, both by sophisticated capacity cut generation. Pessoa et al. (2008) also mention a straight-forward hop-indexed formulation for the asymmetric capacitated vehicle routing problem, which can be traced back almost 50 years to a hop-indexed formulation for the traveling salesman problem by Hadley (1964). Godinho et al. (2008) present several ways to strengthen the hop-indexed formulation. They note that their intent is not to develop efficient solution methods. They credit the branch-and-cut procedure by Lysgaard et al. (2004) for being most efficient. Pessoa et al. (2008) note that since the number of variables and constraints is proportional to the capacity, hop-indexed formulations are only practical for small capacities. This is the case for OWFACL, where the capacity is between 5 and 10. While the straight-forward hop-indexed formulation is comparatively weak (Godinho et al., 2008), it is both transparent and easy to implement, and real-world OWFACL instances can be solved to optimality within reasonable time by using this formulation.

For the Vehicle Routing Problem (VRP), a considerable number of heuristics have been proposed (Toth and Vigo, 2002). Among those, the Clarke and Wright (1964) savings heuristic is probably the best known.

After introducing notation, we compute the optimal cable layout for three real-world OWFs using a hop-indexed formulation. We then adapt the Clarke and Wright savings heuristic to OWFACL. We present the adapted heuristic's results for the three real-world OWFs, showing that its average performance ratio is as good as 1.02. We present two problem extensions arising from real-world OWF cable layouts. We adapt the hop-indexed formulation to one of them, and show that the thus obtained optimal layouts are up to 13% cheaper than the actually installed layouts. We conclude with a summary of the results and plans for future work.

Preliminaries

In order to best relate to existing literature, we use vehicle routing terminology. Given is a graph $G = (V = V_d \cup V_c, E)$, where V is partitioned into the set V_d of depots and the set V_c of clients. The edge set $E \subseteq V^2$ is the set of potential connections between two nodes in V. To simplify presentation, we assume G to be complete. Given are edge costs $c_{ij} \forall \{i,j\} \in E$, a capacity C, and a set $\chi \subset E^2$ such that $\{\{i,j\},\{u,v\}\} \in \chi$ if edges $\{i,j\}$ and $\{u,v\}$ cross each other.

A *route* is composed of an elementary path connecting at most C clients, and an edge connecting one of the path's end nodes to a depot. We use $r_j = (i_0, i_1, i_2, \ldots, i_k)$ to denote a route $r_j = (V_j, E_j)$ comprising nodes $V_j = \{i_0, i_1, \ldots, i_k\} \subseteq V$, where $i_k \in V_d$, and edge set $E_j = \{\{i_0, i_1\}, \ldots, \{i_{k-1}, i_k\}\} \subseteq E$. Every route has a *first* client, which has exactly one neighbor. All other clients on a route have exactly two neighbors. The *last* client has a depot as neighbor. This implies an orientation of routes towards the depots, that is, a route starts at its first client and ends at a depot.

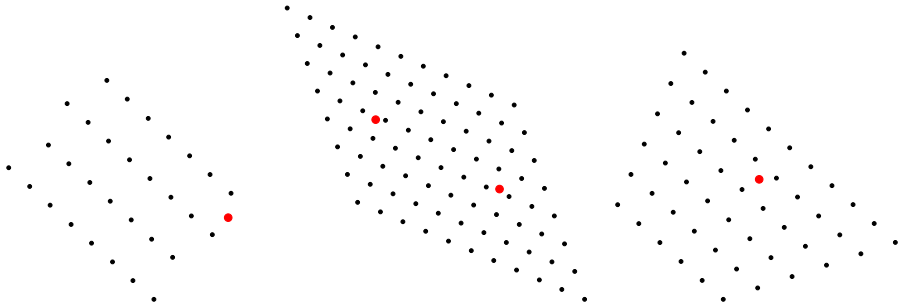
A union of routes such that every client in V_c belongs to exactly one route is called a *routing* (V, R). Since V is implied, we refer to R as the routing. We let $c(R) = \sum_{\{i,j\} \in R} c_{ij}$ be the cost of a routing.

The OWFACL problem or POVRP can then be formulated as: Find a routing R minimizing $c(R)$ and satisfying $R^2 \cap \chi = \emptyset$.

The subsequently presented integer programming formulation and some of our heuristics rely on directed routes and arcs. For any edge set S, we therefore let $A_S = \bigcup_{\{i,j\} \in S} \{(i,j),(j,i)\}$ be the corresponding arc set.

Test Instances

Test instances are given by three OWFs of which the turbines' and transformers' coordinates are publicly available: Barrow OWF (Dong Energy, no date of publication), Sheringham Shoal OWF (Sheringham Shoal, 2012), and Walney 1 OWF (Dong Energy, 2010) and (The Gas and Electricity Markets Authority, 2011), see Fig. 1. They consist of 30, 88, and 51 turbines, respectively, and one, two, and one transformer, respectively. The positions of the turbines of all three farms are defined as points on a grid. The maximum capacity of the installed cables is 10. The trend is for turbines to increase their power generation capacity (Arapogianni et al., 2011), which implies lower cable capacities in terms of turbines per cable. Therefore, we let the capacity range from 5 to 10 for the three farms, yielding 18 test instances. We assume that E is complete, and that c_{ij} equals the geodesic distance between i and j.



(a) Barrow OWF (b) Sheringham Shoal OWF (c) Walney 1 OWF

Figure 1: Test Cases: Real-world OWFs (big nodes mark transformer locations)

A Hop-indexed Formulation with Planarity Constraints

The main idea of a hop-indexed formulation is to let binary variables x_{ij}^h indicate that arc (i, j) belongs to a route, and that i is the h-th node on the route. This facilitates formulating the capacity constraints, since a client j can at most be the C-th node on a route. This leads to the following formulation:

$$x_{ij}^h = \begin{cases} 1 & \text{if } (i,j) \text{ belongs to a route, and } i \text{ is the } h\text{-th node on it} \\ 0 & \text{otherwise.} \end{cases}$$

$$\min \quad \sum_{(i,j)\in A_E} \sum_{h=1}^{C} c_{ij} x_{ij}^h \tag{1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A_E} \sum_{h=1}^{C} x_{ij}^h = 1 \qquad \forall\, i \in V_c \tag{2}$$

$$\sum_{(i,j)\in A_E} x_{ij}^h - \sum_{(j,k)\in A_E} x_{jk}^{h+1} = 0 \qquad \forall\, j \in V_c;\ h = 1, \ldots, C-1 \tag{3}$$

$$\sum_{h=1}^{C} x_{ij}^h + x_{ji}^h + x_{uv}^h + x_{vu}^h \leq 1 \qquad \forall\, \{\{i,j\},\{u,v\}\} \in \chi \tag{4}$$

$$x_{ij}^h \in \{0,1\} \quad \forall\, (i,j) \in A_E,\ h = 1, \ldots, C \tag{5}$$

$$x_{ij}^C = 0 \qquad \forall\, (i,j) \in A_E \cap \{V \times V_c\} \tag{6}$$

Equations (2) assure that every client is left on one arc. Equations (3) count clients on a route: An arc with index h enters j, if and only if an arc with index $h+1$ leaves j. Equations (6) assure that arcs with index C cannot enter a client. Thus, routes cannot visit more than C clients. Equations (4) are the planarity constraints.

Counting the planarity constraints (4) reveals that there are too many to be passed to a solver: For Sheringham Shoal, there are almost 1.5 million planarity constraints. They are also weak: If $\{\{i,j\},\{u,v\}\}$, $\{\{a,b\},\{i,j\}\}$, and $\{\{a,b\},\{u,v\}\} \in \chi$, then at most one of the corresponding arcs can be chosen. Thus, the three respective planarity constraints can be strengthened to $\sum_{h=1}^{C} x_{ab}^h + x_{ba}^h + x_{ij}^h + x_{ji}^h + x_{uv}^h + x_{vu}^h \leq 1$. The strongest planarity constraints are those for maximal sets of arcs crossing each other. Identifying those sets

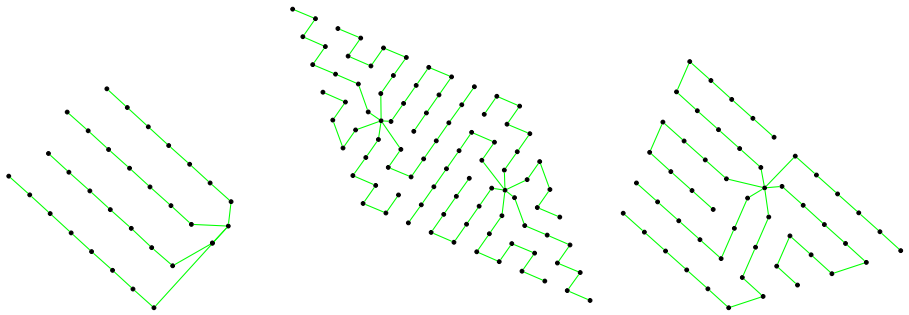
amounts to finding all maximal cliques in the graph (E, χ). Since the problem of finding all maximal cliques in a general graph is NP-hard, this is not a viable strategy.

Consequently, we resort to constraint generation for the planarity constraints. We replace constraints (4) by

$$\sum_{h=1}^{C} x_{ij}^h + x_{ji}^h \leq 1 \,\forall\, (i,j) \in A_E \,. \tag{7}$$

We iterate solving the linear relaxation of (1 - 3, 5 - 7) and adding violated planarity constraints (4) until no violated planarity constraints are found. We then impose the integrality constraints (5) and continue the iterative process, where we in each iteration solve the integer program and add violated planarity constraints (4), until no violated planarity constraints are found. The resulting solution is optimal for (1 - 6).

In this manner, all test instances could be solved (see section on computational results for details). The number of generated planarity constraints is small, between none and nine for all instances. Fig. 2 shows the optimal solutions for the three OWFs for the actually installed maximum cable capacity, which is 8 for Barrow OWF and Sheringham Shoal OWF, and 10 for Walney 1 OWF. Despite the impression given by Fig. 2a, there is no branch in the cable layout, but the cable between the transformer and the southernmost turbine passes the turbine closest to the transformer closely.



(a) Barrow OWF (b) Sheringham Shoal OWF (c) Walney 1 OWF

Figure 2: Optimal Layouts for actually installed Cable Capacities

A Planar Open Savings Heuristic

We adapt the probably best-known VRP heuristic, the Clarke and Wright savings heuristic, to POVRP. The savings heuristic was presented for instances with only one depot d, and constructs routes starting and ending at the depot. It starts from an initial solution, which consists of routes of the form (d, i, d) for all clients $i \in V_c$. The Clarke and Wright savings heuristic thus regards R as a multigraph, which initially contains two copies of every edge $\{i, d\}$ for all clients $i \in V_c$. In every step, the savings heuristic considers *merging* two routes $(d, i, j, \ldots, k, d)$ and $(d, u, v, \ldots, w, d)$ into one route using $\{k, u\}$, that is, into

Parallel Saving (Graph $(V_c \cup \{d\}, E)$, costs $c_{ku} \forall \{k,u\} \in E$, capacity C):

$R \leftarrow \bigcup_{i \in V_c} \{\text{two copies of } \{i,d\}\}$
foreach $\{k,u\} \in E \cap V_c^2$ **do** $s_{ku} = c_{kd} + c_{ud} - c_{ku}$
$S \leftarrow$ sorting of $E \cap V_c^2$ according to decreasing s
repeat
  $\{k,u\} \leftarrow$ next element in S
  if *k and u are in different routes,*
    and $\{k,d\} \in R$ and $\{u,d\} \in R$,
    and the total number of clients in the paths containing k and u
    does not exceed C **then**
      $R \leftarrow R \setminus \{$ one copy each of $\{k,d\}$ and $\{u,d\}\}$
      $R \leftarrow R \cup \{\{k,u\}\}$
until *end of S is reached*
return R

Figure 3: The Clarke and Wright Parallel Savings Heuristic

$(d, i, j, \ldots, k, u, v, \ldots, w, d)$. The saving s_{ku} associated with $\{k,u\}$ is the saving achieved by merging those two routes using $\{k,u\}$. Thus, $s_{ku} = c_{kd} + c_{ud} - c_{ku}$. In every step, the savings heuristic greedily chooses the merge with the highest saving resulting in a route not exceeding capacity.

The savings heuristic exists in a sequential and a parallel version. The sequential version merges into only one route at a time until its capacity is reached. The parallel version considers all incumbent routes for merging. We adapt the parallel savings heuristic, which for reference is given in Fig. 3.

Adapting the Savings Heuristic to POVRP raises two issues: Open routes and planarity.

Open routes take the form $(i, j, \ldots, k, d)$. Thus, the initial solution now consists of routes of the form (i,d) for all clients $i \in V_c$. Merging the two routes $(i, j, \ldots, k, d)$ and $(u, v, \ldots, w, d)$ into $(i, j, \ldots, k, u, v, \ldots, w, d)$ achieves the saving $s_{ku} = c_{kd} - c_{ku}$. Since s_{ku} in general does not equal s_{uk}, savings are associated with arcs instead of edges. Then, a merge using (k,u) requires that $\{k,d\} \in R$, and that u has only one neighbor in R, that is, that k and u are the last and first clients on their respective routes.

Planarity can readily be dealt with by the additional requirement $\{\{k,u\},\{i,j\}\} \notin \chi \forall \{i,j\} \in R$, that is, the routes containing k and u are merged using (k,u) only if $\{k,u\}$ does not cross any edge in R.

Finally, the planar open savings heuristic should work for a set of depots. Thus, in the initial solution, every client is connected to the depot it can be connected to at the least cost. The resulting first Planar Open Savings (POS1) heuristic is given in Fig. 4.

POS1 maintains a feasible solution throughout execution, as do the subsequently presented heuristics. POS1 has time complexity $O(|E| \log |E| + |E||V_c|)$, since sorting A_E has time complexity $O(|E| \log |E|)$, and processing S with checking for crossings has time complexity $O(|E||V_c|)$.

For our test instances, the solutions output by POS1 are on average only 5% more expensive than the optimal solutions (see third column in Tab. 1). Analyzing the solutions for possible improvements of POS1 shows that it for a

POS1 (Graph $(V_c \cup V_d, E)$, costs $c_{ku} \forall \{k,u\} \in E$, capacity C):

1 **foreach** $i \in V_c$ **do** $d_i = \arg\min_{d \in V_d}\{c_{id}\}$
2 $R \leftarrow \bigcup_{i \in V_c}\{\{i, d_i\}\}$
3 **foreach** $(k,u) \in A_E \cap V_c^2$ **do** $s_{ku} = c_{kd_k} - c_{ku}$
4 $S \leftarrow$ sorting of $A_E \cap V_c^2$ according to decreasing s
5 **repeat**
6 $(k,u) \leftarrow$ next element in S
7 **if** *k and u are in different paths,*
8 *and $\{k, d_k\} \in R$,*
9 *and u has only one neighbor in R,*
10 *and the total number of clients in the paths containing k and u*
11 *does not exceed C,*
12 *and $\{k,u\}$ does not cross any edge in R* **then**
13 $R \leftarrow R \setminus \{\{k, d_k\}\} \cup \{\{k,u\}\}$
14 **until** *end of S is reached*
15 **return** R

Figure 4: The first Planar Open Saving Heuristic

couple of test instances generates sub-optimal routes as shown in Fig. 5: After POS1 has added arcs (C1,C2) and (D1,D2) to R, POS1 cannot merge the routes containing them using (C1,D1) or (D1,C1), since neither C1 nor D1 is the last client on its route.

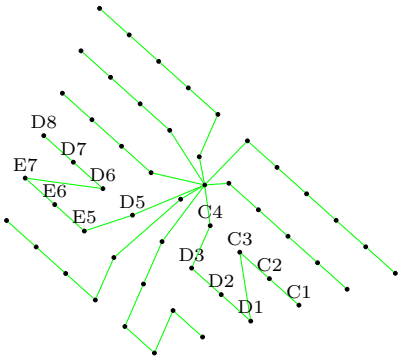


Figure 5: Routing generated by POS1 for Walney 1 OWF with $C = 10$

We consider two strategies to remedy these sub-optimal routes: A local search heuristic to be applied on the solution output by POS1, and adapting POS1 such that it allows merging routes by connecting their first clients.

The second strategy results in the heuristic presented in Fig. 6.

The reinsertions in line 17 are necessary for the following reason: In line 16, POS2 merges the two routes $r_k = (k, \ldots, i, d_i)$ and $r_u = (u, \ldots, w, d_w)$ using (k,u) into the new route $r = (i, \ldots, k, u, \ldots, w, d_w)$. If POS2 had considered a merge using (j,i) while i had two neighbors in R (one of them d_i), the merge using (j,i) had to be discarded. But after the merge using (k,u), client i becomes

POS2 (Graph $(V_c \cup V_d, E)$, costs $c_{ku} \forall \{k,u\} \in E$, capacity C):

```
1  foreach i ∈ V_c do d_i = arg min_{d∈V_d} {c_id}
2  R ← ⋃_{i∈V_c} {{i, d_i}}
3  foreach (k,u) ∈ A_E ∩ V_c^2 do s_ku = c_{kd_k} − c_ku
4  S ← sorting of A_E ∩ V_c^2 according to decreasing s
5  repeat
6      (k,u) ← next element in S
7      if k and u are in different paths,
8        and {k, d_k} ∈ R, or k has only one neighbor in R,
9        and u has only one neighbor in R,
10       and the total number of clients in the paths containing k and u
11       does not exceed C,
12       and {k,u} does not cross any edge in R then
13         if {k, d_k} ∈ R then R ← R \ {{k, d_k}} ∪ {{k,u}}
14         else // k has only one neighbor in R, which is not d_k
15             i ← client in the route containing k with {i, d_i} ∈ R
16             R ← R \ {{i, d_i}} ∪ {(k,u)}
17             re-insert into S all arcs (j,i) that were discarded earlier
               because i had two neighbors in R (d_i being one of them)
18         v ← first client of the merged route
19         w ← last client of the merged route
20         foreach n ∈ V_c do
21             if n has exactly one neighbor in R then
22                 s_vn ← c_{wd_w} − c_vn, and update S accordingly
23 until end of S is reached
24 return R
```

Figure 6: The second Planar Open Saving Heuristic

the first client in the merged route, thus potentially allowing a merge using (j,i). The update in lines 20-22 is due to the change of saving that can be achieved by a merge using an arc (v,n), where v is the first client of the merged route. This merge removes $\{w,d_w\}$ from R, where w is the last client of the merged route, and adds $\{v,n\}$ to R. Thus, the saving that is achieved by this merge changes to $s_{vn} = c_{wd_w} - c_{vn}$.

There are at most $|V_c|$ elements to be reinserted into S in line 17, and at most $|V_c|$ elements of S to be updated in line 22. If S is implemented as a priority queue using a Fibonacci heap, every reinsertion in line 17 takes constant time. Every update in line 20 has amortized time complexity in $O(\log|E|)$: Since the value of s_{vn} may decrease, while S is a find-maximum priority queue, updating s_{vn} amounts to deleting the element (v,n) from S (this has amortized time complexity in $O(\log|E|)$) and to reinserting it with the new value. Thus, POS2 has time complexity in $O(|E|\log|E| + |V_c||E| + |V_c|^2\log|E|) = O(|V_c||E| + |V_c|^2\log|E|)$, which for $|E| \in \Theta(|V|^2)$ is in $O(|E|\log|E| + |V_c||E|)$, the time complexity of POS1.

POS2 outperforms POS1 for all except two of the test instances. The routings generated by POS2 for the test instances are on average only 2.6% more expensive than the optimal routings (see fifth column in Tab. 1). For example,

applying POS2 to Walney 1 OWF with $C = 10$ (Fig. 7) generates a routing which is only 1.6% more expensive than the optimal routing, while the routing generated by POS1 (Fig. 5) is 9.3% more expensive than the optimal routing.

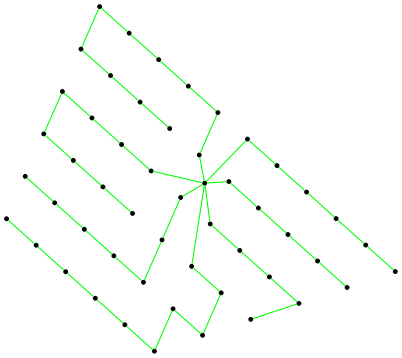


Figure 7: Routing generated by POS2 for Walney 1 OWF with $C = 10$

A Local Search Heuristic for POS1

Another strategy is to locally improve the generated sub-optimal routes. This can be achieved using a 2-exchange-heuristic. Let $r = (i_0, i_1, i_2, \ldots, i_{\ell-1}, i_\ell, \ldots, i_k)$ be a route, where $i_k \in V_d$. We let the search neighborhood of this route comprise all feasible routes $r_\ell = (i_{\ell-1}, \ldots, i_2, i_1, i_0, i_\ell, \ldots, i_k)$, that is, routes generated from r by exchanging edge $\{i_{\ell-1}, i_\ell\}$ in r with $\{i_\ell, i_0\}$, and where the new edge $\{i_\ell, i_0\}$ does not cross any of the edges in R. The improvement achieved by this exchange is $c_{i_{\ell-1},i_\ell} - c_{i_\ell,i_0}$. For every route in a routing, the local search heuristic RouteOpt searches its neighborhood for the maximum improvement, and if it is positive, RouteOpt performs the exchange. This is iterated for every route until its neighborhood does not contain a route with positive improvement, see Fig. 8.

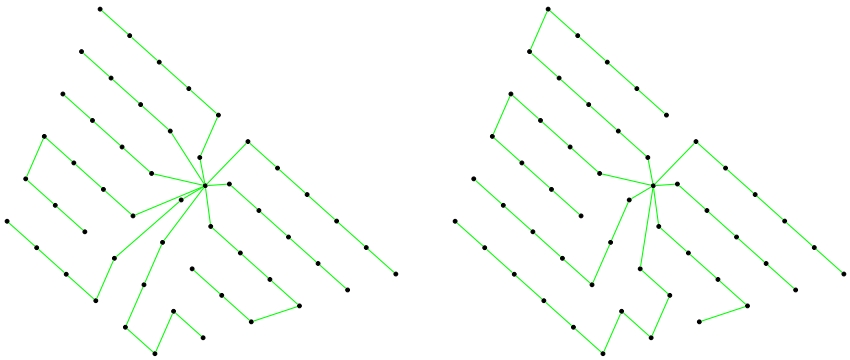
For example, applying RouteOpt to the routing generated by POS1 for Walney 1 OWF with $C = 10$ (Fig. 5), exchanges (C3,D1) with (C1,D1), and subsequently (D3,C4) with (C3,C4), and in the same manner replaces (D6,E7) with

RouteOpt (Graph $(V_c \cup V_d, E)$ with edge costs $c_{ij} \forall \{i,j\} \in E$, feasible routing R):

foreach *route* $r = (i_0, i_1, i_2, \ldots, i_{k-1}, i_k) \in R$ **do**
  repeat
    $s \leftarrow \max_{\ell \in 1,\ldots,k-1} \{c_{i_{\ell-1},i_\ell} - c_{i_\ell,i_0} : \{\{i_\ell, i_0\}, \{i,j\}\} \notin \chi \, \forall \{i,j\} \in R\}$
    if $s > 0$ **then**
      $\ell \leftarrow$ index for which $c_{i_{\ell-1},i_\ell} - c_{i_\ell,i_0} = s$
      $R \leftarrow R \setminus \{\{i_{\ell-1}, i_\ell\}\} \cup \{\{i_\ell, i_0\}\}$
      $r \leftarrow (i_{\ell-1}, \ldots, i_2, i_1, i_0, i_\ell, \ldots, i_k)$
  until *no improvement in* r *is found*

Figure 8: The Local Search Heuristic RouteOpt

(D8,E7) and subsequently (D5,E5) with (D6,D5), see Fig. 9. This improves the routing from being 9.3% more expensive than the optimal routing to being only 6% more expensive.



(a) Routing by POS1 and RouteOpt (b) Routing by POS2 and RouteOpt

Figure 9: Routing generated by applying RouteOpt to the routings generated by POS1 and POS2 for Walney 1 OWF with $C = 10$

Since every client can become the first client of a route at most once, the repeat-loop in RouteOpt is executed at most $|V_c|$ times. Executing the loop has time complexity in $O(C)$ for finding the biggest improvement. Thus, RouteOpt has time complexity in $O(|V_c|C)$ and can be executed after POS1 and POS2 without increasing the overall time complexity. Finally, since POS1 and POS2 for $|E| \in \Theta(|V|^2)$ have the same time complexities, both can be executed without increasing the overall time complexity, and the best resulting layout can be chosen. We call this "best of"-heuristic POS, see Fig. 10.

Computational Results for Test Instances

Computational experiments were performed on a standard personal computer with 2.83GHz processor and 7.7GB of RAM. The integer formulation was implemented in GAMS, and instances were solved using CPLEX 12.3 with default options. All instances were solved to optimality within a few seconds to 20 minutes, except Sheringham Shoal with capacity 10, which was aborted after an hour with relative optimality gap 0.0004. The heuristics were implemented in Java, with the longest run (POS1 for Sheringham Shoal with capacity 5) taking less than 0.06 seconds, and the subsequent longest run of RouteOpt taking 10^{-4} seconds.

POS (Graph $(V_c \cup V_d, E)$ with edge costs $c_{ij} \forall (i,j) \in E$):

1 $R_1 \leftarrow \text{RouteOpt}((V_c \cup V_d, E), c, \text{POS1}((V_c \cup V_d, E), c))$
2 $R_2 \leftarrow \text{RouteOpt}((V_c \cup V_d, E), c, \text{POS2}((V_c \cup V_d, E), c))$
3 **if** $c(R_1) \leq c(R_2)$ **then return** R_1
4 **else return** R_2

Figure 10: The "Best of"-Heuristic POS

For every instance I consisting of an OWF and a capacity, the optimal solution value $c(R_{\min}(I))$ was computed using the hop-indexed formulation. The performance ratio of a heuristic for instance I is then defined as the ratio $\frac{c(R_{\text{heuristic}}(I))}{c(R_{\min}(I))}$.

Table 1 lists the performance ratios of our heuristics for all test instances. The first column identifies the instance by name and C. The second column lists the optimal solution value, rounded to meters. The third and fifth column list the performance ratios of POS1 and POS2, respectively. The fourth and sixth column list the performance ratios of POS1 and POS2 in combination with RouteOpt (RO), respectively. The last column lists the performance ratio of POS. The last line in Tab. 1 lists the average of the respective column.

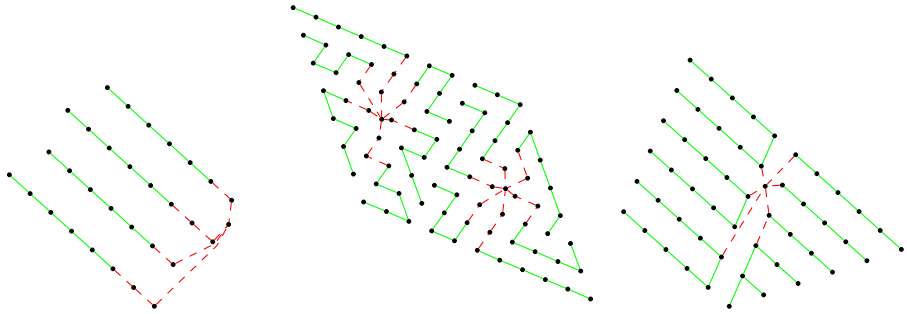
Since both POS2 and RouteOpt were designed to remedy a specific shortcoming of POS1, it is not surprising that RouteOpt only barely improves the routings generated by POS2, and does so only for six of the 18 instances. In general, POS2+RO outperforms POS1+RO. For Barrow OWF with $C = 8$ and 10, both POS1 and POS2 generate the optimal routing. For Barrow OWF with $C = 6, 7$, and 9, POS1 (together with RouteOpt) and POS2 generate the same routing. POS1+RO outperforms POS2+RO only for two of the remaining 13 instances. The combined heuristic POS has an average performance ratio as good as 1.02.

Instance	opt	Performance Ratio				
		POS1	POS1+RO	POS2	POS2+RO	POS
Barrow						
$C = 5$	20739	1.01347	1.01347	1.00434	1.00434	1.00434
$C = 6$	18375	1.02992	1.02987	1.02987	1.02987	1.02987
$C = 7$	17781	1.00304	1.00304	1.00304	1.00304	1.00304
$C = 8$	16566	1.00000	1.00000	1.00000	1.00000	1.00000
$C = 9$	16553	1.01006	1.01006	1.01006	1.01006	1.01006
$C = 10$	16317	1.00000	1.00000	1.00000	1.00000	1.00000
Sheringham Shoal						
$C = 5$	64828	1.04353	1.03147	1.03963	1.03843	1.03147
$C = 6$	62031	1.07176	1.0459	1.04139	1.04014	1.04014
$C = 7$	60667	1.08304	1.05787	1.05574	1.05574	1.05574
$C = 8$	59836	1.07289	1.04737	1.02343	1.02214	1.02214
$C = 9$	59274	1.07149	1.04573	1.0153	1.01399	1.01399
$C = 10$	58960	1.0772	1.0513	1.02071	1.01939	1.01939
Walney 1						
$C = 5$	43539	1.05125	1.05125	1.04293	1.04293	1.04293
$C = 6$	41587	1.05282	1.05282	1.03755	1.03755	1.03755
$C = 7$	40789	1.05974	1.02796	1.04776	1.04776	1.02796
$C = 8$	40242	1.07414	1.04193	1.03045	1.03045	1.03045
$C = 9$	39752	1.08738	1.05477	1.04204	1.04204	1.04204
$C = 10$	39541	1.09318	1.0604	1.01603	1.01285	1.01285
average		1.04972	1.03474	1.02557	1.02504	1.02355

Table 1: Performance of POS1 and POS2 (with RouteOpt), and of POS

Problem Extensions

In this section, we present two extensions of OWFACL. Figure 11 shows the layouts actually installed in the three OWFs. In all three farms, two cable types (dashed red and green) are used. In Walney 1, the layout does not consist of routes, but allows branching.



(a) Barrow OWF (b) Sheringham Shoal OWF (c) Walney 1 OWF

Figure 11: Actually Installed Layouts

Branching

It is possible to let cables branch at a turbine. Every branch induces the cost of additional electronic equipment (so-called switchgear). OWFACL with branching amounts to a Capacitated Minimum Spanning Tree problem (Voß, 2009) with node degree dependent costs (Gouveia and Moura, 2010).

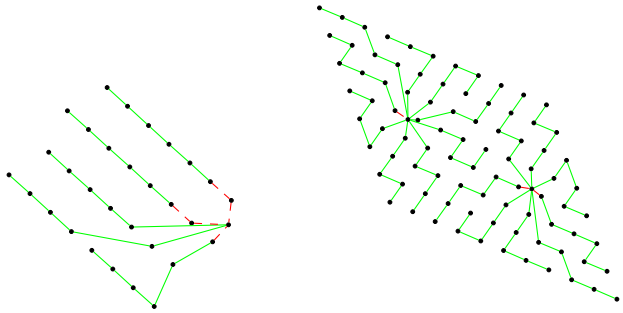
Cable Choice

Cable types are distinguished by capacity and cost: The higher a cable's capacity, the more expensive it is. If there are two cable types with capacities $C < Q$ and costs $c < q$ available, optimizing the cable layout can readily be achieved by changing the objective function (1) of the hop-indexed VRP formulation into

$$\min \sum_{(i,j)\in A} \left(\sum_{h=1}^{C} c_{ij} x_{ij}^h + \sum_{h=C+1}^{Q} q_{ij} x_{ij}^h \right) \tag{8}$$

Based on the actually installed layouts, we assume that the two available cable types have capacities $C = 5$, and $Q = 8$ for Barrow and Sheringham Shoal OWFs, and $Q = 10$ for Walney 1 OWF. According to (Dong Energy, no date of publication), the cables installed at Barrow OWF have diameters 120mm and 300mm, which implies $c = 80\frac{€}{m}$ and $q = 140\frac{€}{m}$ (Fosse, 2010). According to (LORC, 2011a), the cables installed at Sheringham Shoal OWF have diameters 185mm and 400mm, which implies $c = 110\frac{€}{m}$ and $q = 180\frac{€}{m}$ (Fosse, 2010). With this input data, we solve formulation (8, 2 - 6) iteratively in the same manner as formulation (1 - 6). The instances for Barrow OWF and Sheringham Shoal were solved to optimality within 16 seconds and 8 minutes, respectively. The resulting layouts are shown in Fig. 12. The costs of these layouts are 94% and

87% of the costs of the actual installed layouts for Barrow OWF and Sheringham Shoal OWF, respectively. With the given data, this amounts to total savings of €134080 and €1072720 for Barrow OWF and Sheringham Shoal OWF.



(a) Barrow OWF (b) Sheringham Shoal OWF

Figure 12: Optimal Layouts with two Cable Types

According to (LORC, 2011b), the cables installed at Walney 1 OWF have diameters 150mm and 500mm. For the latter we could not find cable costs, and since we neither found information on switchgear costs, a comparison of the costs of the optimal and the installed layouts is not possible for this OWF.

Conclusions

We have presented the Offshore Wind Farm Array Cable Layout Problem, which can be characterized as a Planar Open Vehicle Routing Problem with unit demands. We have shown how real-world instances can be solved by planarity constraint generation using a hop-indexed formulation. We have presented the heuristics POS1 and POS2 with time complexities in $O(|V_c|^2 \log |E| + |E||V_c|)$ and the local search heuristic RouteOpt with time complexity in $O(|V_c|C)$. By combining all these heuristics into the heuristic POS, we can generate routings on average only 2% more expensive than the optimal routings.

We have presented two problem extensions, namely layouts allowing branching and choosing cable types. We have shown how the hop-indexed formulation can be readily adapted to choose cable types, and that the cost saving of the optimal solutions compared with the actually installed layouts are significant.

Future work will be concerned with exact and heuristic solutions for layouts with branching, and heuristics for layouts with cable choice.

References

Arapogianni A, Moccia J, Williams D, Phillips J, (2011). *Wind in our Sails.* Technical Report. European Wind Energy Association.

Clarke G, Wright J, (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* **12**: 568–581.

Dong Energy, (2010). `http://www.dongenergy.com/Walney/News/data/Documents/WOW_I_grid.pdf`, accessed 15 March 2013.

Dong Energy, (no date of publication). http://www.lorc.dk/handlers/dh.ashx?id=32, accessed 15 March 2013

Fosse T, (2010). *Optimizing the Infield Cable Layout in Offshore Wind Farms with Network Design Algorithms.* Master thesis. Bergen University College.

Godinho, M. T., Gouveia, L., Magnanti, T. L., 2008. Combined route capacity and route length models for unit demand vehicle routing problems. *Discrete Optimization* 5 (2), 350–372.

Gouveia L, Moura P, (2010). Spanning trees with node degree dependent costs and knapsack reformulations. *Electronic Notes in Discrete Mathematics* **36**: 985 – 992.

Hadley, G., 1964. *Nonlinear and dynamic programming*, Addison-Wesley Publishing Company.

Letchford A N, Lysgaard J, Eglese R W, (2007). A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society* **58**: 1642–1651.

Li F, Golden B, Wasil E, (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Opererations Research* **34**: 2918–2930.

LORC, (2011a). http://www.lorc.dk/offshore-wind-farms-map/sheringham-shoal, accessed 15 March 2013.

LORC, (2011b). http://www.lorc.dk/offshore-wind-farms-map/walney-1, accessed 15 March 2013.

Lumbreras S, Ramos A, (2012). Offshore wind farm electrical design: a review. *Wind Energy* doi:10.1002/we.1498.

Lysgaard, J, Letchford, A N, Eglese, R W, 2004, A new branch-and-cut algorithm for the capacitated vehicle routing problem, *Mathematical Programming*, Springer, **100**: 423–445,

Pessoa A, Poggi de Aragão M, Uchoa E, (2008). Robust branch-cut-and-price algorithms for vehicle routing problems, in: Golden B, Raghavan S, Wasil E (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. *Operations Research/Computer Science Interfaces*, Springer US, **43**: 297–325.

Réthoré, P.-E., Fuglsang, P., Larsen, G. C., Buhl, T., Larsen, T. J., Madsen, H. A., 2011. Topfarm: Multi-fidelity optimization of offshore wind farm. *The 21st International Offshore (Ocean) and Polar Engineering Conference, ISOPE-2011,*. pp. 19–24.

Sariklis D, Powell S, (2000). A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society* , 564–573.

Sheringham Shoal, (2012). http://www.scira.co.uk/construction/foundationsmap.php, accessed 15 March 2013.

Subramanian A, (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. PhD. thesis, Universidade Federal Fluminense.

The Gas and Electricity Markets Authority, (2011). `http://www.ofgem.gov.uk/Networks/offtrans/rott/w1/Documents1/NOTICE_UNDER_SECTION_8A_(W1).pdf`, accessed 15 March 2013.

Toth P, Vigo D, (2002). *The vehicle routing problem*. Society for Industrial and Applied Mathematics.

Voß S, (2009). Capacitated minimum spanning trees, in: Floudas C A, Pardalos P M (Eds.), *Encyclopedia of Optimization*. Springer US, 347–357.