A savings procedure based construction heuristic for the offshore wind cable layout optimization problem

Sunney Fotedar (B.Eng. Mechanical)
MSc. Candidate in Energy
Department of Informatics, University of Bergen, Norway
sunney.fotedar@student.uib.no
2018
Supervisors: Prof. Dag Haugland (UiB) and Ahmad Hemmati, PhD.
Table of Content

INTRODUCTION
• Offshore Wind Cable Layout (OWCL) Optimization
• Problem Statement and Assumptions
• Constraints: Node crossing/cable crossing, obstacles and out-degree
• Features: Parallel cables and branching
• MILP model used for benchmarking heuristic solutions

HEURISTIC
• Basic idea
• Pseudocode (Esau-Williams)
• Ideas to tackle cable crossing
• Ideas to identify node crossing
• Pseudocode (Obstacle-Aware Esau Williams)

Experimental Results and Modified Algorithm
• Initial results from the Modified Esau-Williams (Wind farms: Walney 1, Walney 2 and Barrow)
• Parametrization and introducing a shape factor
• Improved results
• From construction heuristic to Meta-Heuristic: Future activities (Very large Neighborhood search (VLNS) and GRASP)
# Table of Content

## INTRODUCTION
- Offshore Wind Cable Layout (OWCL) Optimization
- Problem Statement and Assumptions
- Constraints: Node crossing/cable crossing, obstacles and out-degree
- Features: Parallel cables and branching
- MILP model used for benchmarking heuristic solutions

## HEURISTIC
- Basic idea
- Pseudocode (Esau-Williams)
- Ideas to tackle cable crossing
- Ideas to identify node crossing
- Pseudocode (Obstacle-Aware Esau Williams)

## Experimental Results and Modified Algorithm
- Initial results from the Modified Esau-Williams (Wind farms: Walney 1, Walney 2 and Barrow)
- Parametrization and introducing a shape factor
- Improved results
- From construction heuristic to Meta-Heuristic: Future activities (Very large Neighborhood search (VLNS) and GRASP)
Offshore Wind Cable Layout Optimization

- Offshore wind or inter-array cable layout (OWCL) optimization problem is a NP hard problem.
- There is similarity between OWCL and capacitated minimum spanning tree (CMST) problem with unit demand which has also been proved to be NP hard (Papadimitriou, 1978).
- With increasing number of turbine nodes and additional restricted areas in the wind farm, exact methods in solving large instances become inefficient.
- Due to the inefficiencies of the exact methods in solving large instances, heuristics can be used to attain good and feasible solutions.
- Construction, improvement and hybrid heuristics are classical heuristics exploring a limited search space as opposed to large search space in metaheuristics, but using some unique strategies can be used to attain small optimality gap even with classical approaches.
Offshore wind cables

Each node must be connected to one of the substations

Image Source: Bauer et al, 2014
Problem Statement and Assumption

**Problem:**

**Input:**
1. Location of the turbines and substations
2. Location of the restricted areas and obstacles in the sea-bed
3. Cable capacity (maximum power flow or number of turbines allowed on a single cable)

Output: Minimum cable length layout such that there is a unique path from each turbine to one of the substations

**Constraints:**
1. Cable crossing/Node crossing not allowed
2. Cable capacity must be satisfied
3. Outdegree of each turbine is one (no splitting of power cables)

**Assumption:**
Cable cost is directly proportional to the length of the cables and does not depend on any other parameter.

This is similar to a **capacitated minimum spanning tree problem (NP hard)** with some additional constraints.
**Problem Statement and Assumption**

**Problem:**

**Input:**
1. Location of the turbines and substations
2. Location of the restricted areas and obstacles in the sea-bed
3. Cable capacity (maximum power flow or number of turbines allowed on a single cable)

Output: Minimum cable length layout such that there is a unique path from each turbine to one of the substations

**Constraints:**
1. Cable crossing/Node crossing not allowed
2. Cable capacity must be satisfied
3. Outdegree of each turbine is one (no splitting of power cables)

**Assumption:**
Cable cost is directly proportional to the length of the cables and does not depend on any other parameter.

This is similar to a **capacitated minimum spanning tree problem (NP hard)** with some additional constraints.
Constraint 1: Cable crossing and Node crossing

The main reasons behind such a constraint are:
1. Need for expensive bridge structure
2. Thermal interference between the two cables results in reducing the cable capacity
3. In case of failure of one of the cable both the cables are affected while repairing

Image Source: Fischetti et al 2016
Constraint 2: Power cables cannot be splitted

The out-degree of each turbine node must be one. However, in-degree can be more which is referred to as branching.
Constraint 3: Restricted areas

- Direct links are sometimes not possible due to restricted areas in the sea-bed.
- Number of Steiner nodes is a design parameter and can be more than the extreme points of the convex hull.
- We are making an assumption that any concave and convex restricted area can be represented by a convex hull without compromising on optimality.
Constraint 3: Restricted areas

- Direct links are sometimes not possible due to restricted areas in the sea-bed.
- Number of Steiner nodes is a design parameter and can be more than the extreme points of the convex hull.
- We are making an assumption that any concave and convex restricted area can be represented by a convex hull without compromising on optimality.

- Steiner nodes/optional nodes
- Turbine nodes
- Substation node
- Convex hull around the obstacle
- Cables
- Restricted area
Allowed: Branching and parallel cables

Both branching and parallel cables provide flexibility to the final layout and may lead to reduction in the total cable length.

Source: Klein et al 2017

Source: Bauer et al
Table of Content

INTRODUCTION
• Offshore Wind Cable Layout (OWCL) Optimization
• Problem Statement and Assumptions
• Constraints: Node crossing/cable crossing, obstacles and out-degree
• Features: Parallel cables and branching
• MILP model used for benchmarking heuristic solutions

HEURISTIC
• Basic idea
• Pseudocode (Esau-Williams)
• Ideas to tackle cable crossing
• Ideas to identify node crossing
• Pseudocode (Obstacle-Aware Esau Williams)

Experimental Results and Modified Algorithm
• Initial results from the Modified Esau-Williams (Wind farms: Walney 1, Walney 2 and Barrow)
• Parametrization and introducing a shape factor
• Improved results
• From construction heuristic to Meta-Heuristic: Future activities (Very large Neighborhood search (VLNS) and GRASP)
Basic idea behind the heuristic

- **Esau-Williams’** heuristic is a well known heuristic for the capacitated minimum spanning tree problem.
- Start with a costly, feasible star layout
- In each iteration remove one link connecting the non-root node with the root node (substation node) resulting in cost saving.

Although CMST and cable layout problems are quite similar but there are additional constraints which are to be satisfied in the offshore wind cable layout problem.
Basic idea behind the heuristic

- **Esau-Williams’** heuristic is a well known heuristic for the capacitated minimum spanning tree problem.
- Start with a costly, feasible star layout
- In each iteration remove one link connecting the non-root node with the root node (substation node) resulting in cost saving.

Although CMST and cable layout problems are quite similar but there are additional constraints which are to be satisfied in the offshore wind cable layout problem.
Pseudocode of Esau-Williams’ heuristic

Algorithm 1 Esau Williams

Data: G=(V, A, c, K)
Result: M: set of |V| – 1 arcs spanning G

M ← φ
for node i ∈ V\0 do
    M ← M ∪ {(0, i)}; // set of links in spanning tree
    X_i ← {i}
    R_i = .1; // any value except 0
end

while (∃i ∈ V : R_i > 0) do
    for node i ∈ V do
        j(i) ← nearest neighbouring node; // using (2.2.9)
        compute R_i; // Reduction value using (2.2.10)
    end

    find i* ∈ argmax_i∈V R_i
    if R_{i*} > 0 then
        M ← M ∪ {(j(i*), i*)}
        M ← M\{(0, i*)}
        X_{j(i*)} ← X_{i*} ← X_{i*} ∪ X_{j(i*)}
    end
end

return M

V: set of vertices
A: A ⊆ (V × (V\{0}))
0: root node
c: cost of the arcs
K: cable capacity
R_i: reduction function value of node i
X_i: connected component containing node i

S(i) = \{j ∈ V\0 : j ∈ X_i, (j, i) ∈ A, |X_i| + |X_j| ≤ K\} (2.2.8)

\[ j(i) = \begin{cases} \arg\min_{j\in V\setminus\{0\}} [c_{jH} : j \in S(i)], & S(i) \neq \emptyset \\ 0, & S(i) = \emptyset \end{cases} \] (2.2.9)

\[ R_i = \begin{cases} c_{0i} - \min_{j \in S(i)} [c_{jH}], & S(i) \neq \emptyset \\ 0, & S(i) = \emptyset \end{cases} \] (2.2.10)
Pseudocode of Esau-Williams’ heuristic

Algorithm 1 Esau Williams
Data: G=(V, A, c, K)
Result: M: set of |V| – 1 arcs spanning G

M ← φ
for node i ∈ V \ 0 do
  M ← M ∪ {(0, i)} ; // set of links in spanning tree
  X_i ← {i}
  R_i = .1 ; // any value except 0
end
while (∃ i ∈ V : R_i > 0) do
  for node i ∈ V do
    j(i) ← nearest neighbouring node ; // using (2.2.9)
    compute R_i ; // Reduction value using (2.2.10)
  end
  find i* ∈ argmax_{i∈V} R_i
  if R_{i*} > 0 then
    M ← M ∪ {(j(i*), i*)}
    M ← M \ {(0, i*)}
    X_{j(i*)} ← X_{i*} − X_{i*} ∪ X_{j(i*)}
  end
end
return M

Reduction function value (R_i) at each non-root node is the difference of the cost of the central link with the root node and cost of forming a link with the nearest feasible connected component (satisfying the cable capacity limitation)

V: set of vertices
A: A ⊆ (V × (V\{0}))
0: root node
c: cost of the arcs
K: cable capacity
R_i: reduction function value of node i
X_i: connected component containing node i
Idea to tackle cable crossing

Non-crossing procedure and Dijkstra are used subsequently to identify shortest feasible path between two nodes $i_0$ and $i_n$.

```plaintext
DJ ← Dijkstra(G, i_0, i_n)  /* shortest path between i_0 and i_n */
for each arc $(i_k, i_{k+1})$ in the shortest path DJ do
    if Non-Crossing(IntersectionArray, $i_k, i_{k+1}$) == FALSE
        then
            $c_{i_ki_{k+1}} ← ∞$;  // any high value
    end
end
```

Continues until a shortest feasible (non-crossing) path is found between $i_0$ and $i_n$.

So, the basic idea is that once we have identified the two turbine nodes to be connected using the max reduction function value, we try to use the above idea to find the shortest non-crossing path between them.
Obstacle-Aware Esau Williams Heuristic (1/2)

Algorithm 2 Obstacle Aware Esau Williams

Data: G=(V, A, c, K), L, C, PointArray
Result: M: set of rooted trees spanning (T ∪ S)

L2 ← φ, M ← φ

while any(R) do

/* #1 */
FirstTime ← 0
while FirstTime ≠ 1 do

/* #2 */
for node i ∈ T do

compute Ri; // Reduction value using (2.2.10)
minIndex[i] ← j(i); // equation (2.2.9)
end

for each line segment e in L2 do

IntersectionArray.add(e)
end

for each line segment s in L do

IntersectionArray.add(s)
end

i_0 ← i* ∈ argmax_{i∈T} R_i
i_n ← minIndex[i*]; // Now we have arc (i_0, i_n) to be checked in pre-processing stage
crossingSwitch ← TRUE

- L: stores line segments related to the obstacles
- PointArray: stores coordinates of all the nodes
- L2: stores the arcs/line segments formed during the procedure
- IntersectionArray: stores both L2 and L
- While loop #1: continues unless all the reduction values become zero
- While loop #2: continues unless the node with highest reduction values gets linked with another node
Obstacle-Aware Esau Williams Heuristic(1/2)

We have selected node $i_0$ having the maximum reduction function value and its nearest node $i_n$. Now, in pre-processing stage the shortest feasible path between them is searched which may or may not be a direct arc

---

Algorithm 2 Obstacle Aware Esau Williams

**Data:** $G=(V, A, c, K), L, C, PointArray$

**Result:** $M$: set of rooted trees spanning $T \cup S$

$L2 \leftarrow \emptyset$, $M \leftarrow \emptyset$

**while any(R) do**

/* #1*/

FirstTime $\leftarrow 0$

**while FirstTime$\neq 1$ do**

/* #2*/

for node $i \in T$ do

compute $R_i$; // Reduction value using (2.2.10)

$minIndex[i] \leftarrow j(i)$; // equation (2.2.9)

end

for each line segment $e$ in $L2$ do

IntersectionArray.add($e$)

end

for each line segment $s$ in $L$ do

IntersectionArray.add($s$)

end

$i_0 \leftarrow i^* \in \text{argmax}_{i \in T} R_i$

$i_n \leftarrow minIndex[i^*]$; // Now we have arc $(i_0, i_n)$ to be checked in pre-processing stage

crossingSwitch $\leftarrow$ TRUE

- $L$: stores line segments related to the obstacles
- **PointArray**: stores coordinates of all the nodes
- $L2$: stores the arcs/line segments formed during the procedure
- **IntersectionArray**: stores both $L2$ and $L$
- **While loop #1**: continues unless all the reduction values become zero
- **While loop #2**: continues unless the node with highest reduction values gets linked with another node
Obstacle-Aware Esau Williams Heuristic (2/2)

```c
/* #3 */
while crossingSwitch == TRUE do
    FirstTime ← FirstTime + 1
    crossingSwitch ← FALSE
    DJ ← Dijkstra(G, i_0, i_n) /* shortest path between i_0 and i_n */
    for each arc (i_k, i_{k+1}) in the shortest path DJ do
        if Non-Crossing(IntercetionArray, i_k, i_{k+1}) == FALSE then
            c_{i_ki_{k+1}} ← ∞; // any high value
            crossingSwitch ← TRUE
        end
    end
    if !crossingSwitch&&FirstTime ≠ 1 then
        C[i_0][i_n] ← c_{i_0i_1} + c_{i_1i_2} + ··· + c_{i_{n-1}i_n}; // exit loop #3
    end
end
post node joining steps and updates of connected components, L2, M
end
formng shortest feasible paths from an active node in each connected component to the substation
return M
```

Pre-processing stage
Non-Crossing procedure’s output

i. True

ii. True

iii. False

iv. True

v. True

vi. True
Non-Crossing procedure’s output

Note: Non-crossing procedure always assesses pair of line segments (one from IntersectionArray and one of the edge in the shortest path given by Dijkstra)
Challenge: Non-Crossing procedure is unable to identify node crossing

- i. False
- ii. True
- iii. False
- iv. True
- v. True
- vi. True
Challange: Non-Crossing procedure is unable to identify node crossing
Challenge: Non-Crossing procedure is unable to identify node crossing
Challenge: Non-Crossing procedure is unable to identify node crossing
Solution(1/4): Add new line segments such that node crossings are detected by Non-crossing procedure.
Solution(2/4): Add new line segments such that node crossings are detected by Non-crossing procedure.
Solution(3/4): Add new line segments such that node crossings are detected by Non-crossing procedure
Solution(4/4): Add new line segments such that node crossings are detected by Non-crossing procedure.

Feasible Connection:

(i) False
(ii) True
(iii) False
(iv) True
(v) True
(vi) True
Solution(1/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post partitioning step
Solution (1/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Post partitioning step

Output from 1st part of the algorithm
Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post joining step

Dijkstra
Graham scan

Convex hull of the connected component
Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post joining step

Dijkstra
Graham scan
Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components → Post joining step → Output from 1st part of the algorithm

Dijkstra
Graham scan
Clique formation
Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post joining step

Dijkstra
Graham scan
Clique formation

Feasible Connection

Assumption: All the nodes are in the convex hull of their own connected component and not in the convex hull of others’
Table of Content

INTRODUCTION

• Offshore Wind Cable Layout (OWCL)Optimization
• Problem Statement and Assumptions
• Constraints: Node crossing/cable crossing, obstacles and out-degree
• Features: Parallel cables and branching
• MILP model used for benchmarking heuristic solutions

HEURISTIC

• Basic idea
• Pseudocode (Esau-Williams)
• Ideas to tackle cable crossing
• Ideas to identify node crossing
• Pseudocode (Obstacle-Aware Esau Williams)

Experimental Results and Modified Algorithm

• Initial results from the Modified Esau-Williams (Wind farms: Walney 1, Walney 2 and Barrow)
• Parametrization and introducing a shape factor
• Improved results
• From construction heuristic to Meta-Heuristic: Future activities (Very large Neighborhood search (VLNS) and GRASP)
Experimental Results-1

Existing Model:
- We have compared our results to the optimal solutions attained from an existing MILP model developed by our colleague Arne Klein, UiB, Norway.
- The model presented in [Klein and Haugland, 2017] is implemented using CPLEX 12 Python 3.4 API. All the experiments were carried out on a fast computer - Intel Xenon with 72 logical cores and 256GB RAM.
- The experiments were carried out for Walney 1, Walney 2, Barrow wind farms and for different cable capacities.

Developed Heuristic:
- All the experiments involving the heuristics (Obstacle Aware Esau-Williams) in this work are carried out on a personal computer using 2.5 GHz Intel Core i5 processor and 4GB RAM.
- Programming language used is Java and without use of any commercial solver.
- The ambition of the first version of the obstacle-aware heuristic is to find good, feasible solutions with less optimality gap \[\frac{\text{cost(heuristic)}}{\text{cost(optimal solution)}}\]
## Experimental Results-2

<table>
<thead>
<tr>
<th>K</th>
<th>Barrow(T=30)</th>
<th>Walney-1(T=51)</th>
<th>Walney-2(T=51)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact</td>
<td>Alg 2</td>
<td>gap</td>
</tr>
<tr>
<td>2</td>
<td>36990</td>
<td>38115</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>23208</td>
<td>23243</td>
<td>1.00</td>
</tr>
<tr>
<td>5</td>
<td>20691</td>
<td>21815</td>
<td>1.05</td>
</tr>
<tr>
<td>6</td>
<td>18374</td>
<td>20980</td>
<td>1.14</td>
</tr>
</tbody>
</table>

K= cable capacity, Alg2 = Obstacle-Aware Esau Williams

Walney 1 final layout for K=6
Experimental Results-3

- There is a large optimality gap for Walney 2
- The partitioning of the turbine nodes leads to extremely long paths connecting connected components to the substation
- For example, the connected component containing nodes 45, 46, 47, 48, 49, 39 is linked with the substation using a long path 45->38->27->51
There is a large optimality gap for Walney 2.

The partitioning of the turbine nodes leads to extremely long paths connecting connected components to the substation.

For example, the connected component containing nodes 45, 46, 47, 48, 49, 39 is linked with the substation using a long path 45->38->27->51.
Ideas/Activities to reduce the opt. gap

- Modifying the reduction function and the algorithm such that radial topologies are encouraged and thus, long paths to the substation are avoided.

- Using a multi-exchange large neighbourhood search for finding the locally optimal solution.
Introducing weight parameter in reduction function

\[ S(i) = \{ j \in V \setminus 0 : j \notin X_i, (j, i) \in A, |X_i| + |X_j| \leq K \} \quad (2.2.11) \]

\[ j(i) = \begin{cases} 
\text{one of the } j \in \arg\min_{j \in V \setminus \{0\}} \{ c_{ji} + \frac{W}{1000} \times c_{0j} : j \in S(i), W \in [1,1000], W \in \mathbb{Z} \}, & S(i) \neq \emptyset \\
0, & S(i) = \emptyset 
\end{cases} \quad (2.2.12) \]

\[ R_i = \begin{cases} 
c_{0i} - \min \{ c_{ji} + \frac{W}{1000} \times c_{0j} + : j \in S(i) \}, & S(i) \neq \emptyset \\
0, & S(i) = \emptyset 
\end{cases} \quad (2.2.13) \]
Introducing weight parameter in reduction function

\[ S(i) = \{ j \in V \setminus 0 : j \notin X_i, (j, i) \in A, |X_i| + |X_j| \leq K \} \quad (2.2.11) \]

\[ j(i) = \begin{cases} 
\text{one of the } j \in \arg \min_{j \in V \setminus \{0\}} \{ c_{ji} + \frac{W}{1000} \times c_{0j} : j \in S(i), W \in [1, 1000], W \in \mathbb{Z} \}, & S(i) \neq \emptyset \\
0, & S(i) = \emptyset 
\end{cases} \quad (2.2.12) \]

\[ R_i = \begin{cases} 
c_{0i} - \min\{ c_{ji} + \frac{W}{1000} \times c_{0j} : j \in S(i) \}, & S(i) \neq \emptyset \\
0, & S(i) = \emptyset 
\end{cases} \quad (2.2.13) \]

As the value of weight parameter W increases, turbine nodes closer to the substation will be preferred.
Results from exact, obstacle aware Esau Williams and algorithm with weight parameter

<table>
<thead>
<tr>
<th>Wind Farm</th>
<th>Exact</th>
<th>Obstacle-Aware</th>
<th>Parametric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>value</td>
<td>gap</td>
<td>value</td>
</tr>
<tr>
<td>K=6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walney 1</td>
<td>41418</td>
<td>43858</td>
<td>42580</td>
</tr>
<tr>
<td>Barrow</td>
<td>18374</td>
<td>20980</td>
<td>18900</td>
</tr>
<tr>
<td>Walney 2</td>
<td>52981</td>
<td>63568</td>
<td>53214</td>
</tr>
<tr>
<td>K=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walney 1</td>
<td>43420</td>
<td>44444</td>
<td>43498</td>
</tr>
<tr>
<td>Barrow</td>
<td>20691</td>
<td>21815</td>
<td>21105</td>
</tr>
<tr>
<td>Walney 2</td>
<td>56904</td>
<td>62739</td>
<td>57816</td>
</tr>
<tr>
<td>K=4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walney 1</td>
<td>47411</td>
<td>49534</td>
<td>48396</td>
</tr>
<tr>
<td>Barrow</td>
<td>232208</td>
<td>23243</td>
<td>23243</td>
</tr>
<tr>
<td>Walney 2</td>
<td>63496</td>
<td>73374</td>
<td>63579</td>
</tr>
</tbody>
</table>
Improved result for Walney 2

Walney 2 (Modified Esau Williams Algorithm-Parametric)

Walney 2 (Modified Esau Williams Algorithm)
Change in cable length with weight parameter
Ideas/Activities to reduce the opt. gap

- Modifying the reduction function and the algorithm such that radial topologies are encouraged and thus, long paths to the substation are avoided

- Using a multi-exchange large neighbourhood search for finding the locally optimal solution
Questions?

Thank You!

*Project is supported by Hordaland fylkeskommune.*