# 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.

Deep Wind Conference 2018, Trondheim, Norway

# 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

# 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 miniumum 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

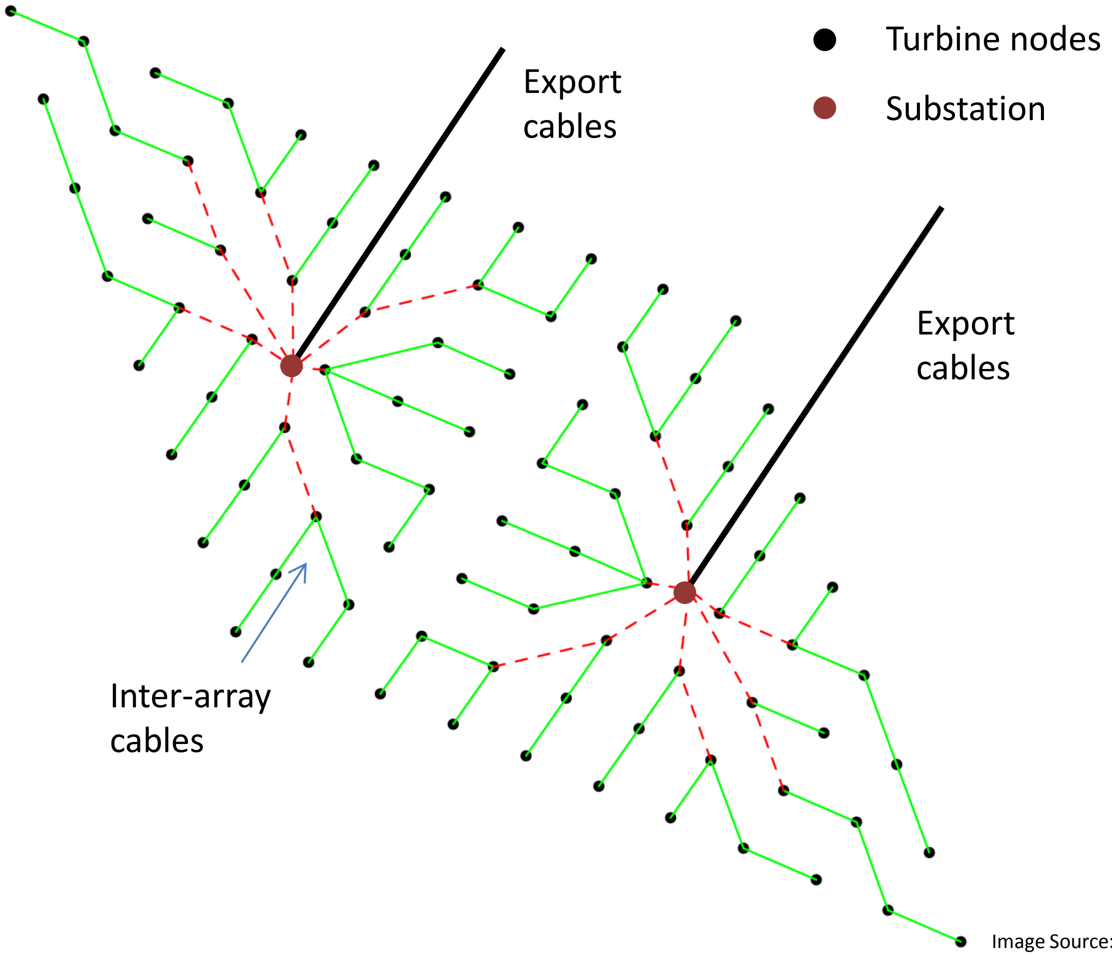



Image Source: Bauer et al, 2014

**Each node must be connected to one of the substation**

# Problem Statement and Assumption

**<u>Problem</u>** :

**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 substation

**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 doesnot 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** :

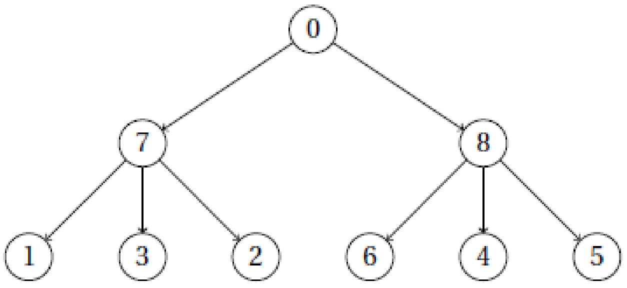
**Input**:

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 substation

**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)



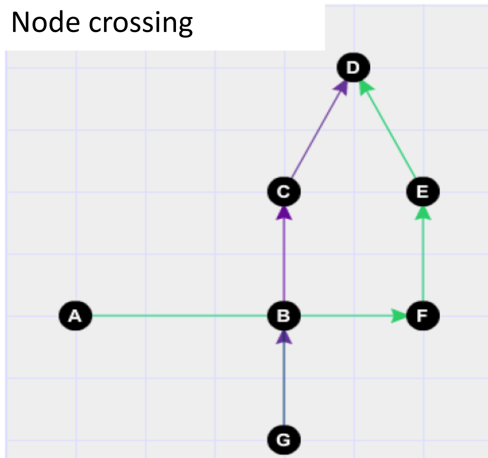

Capacity = 4

**Assumption**:

Cable cost is directly proportional to the length of the cables and doesnot 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



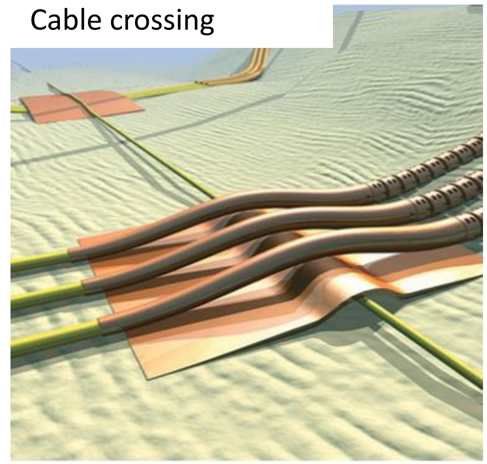
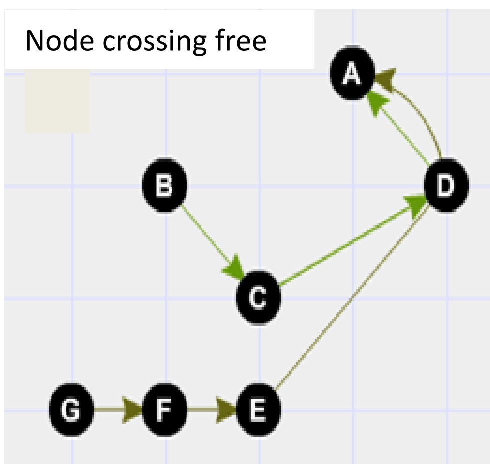



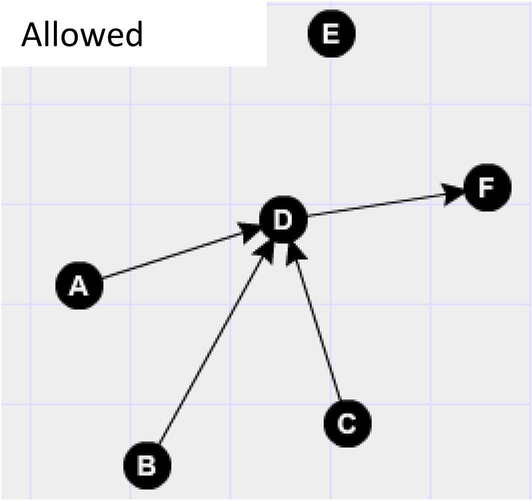

Image Source: Fischetti et al 2016

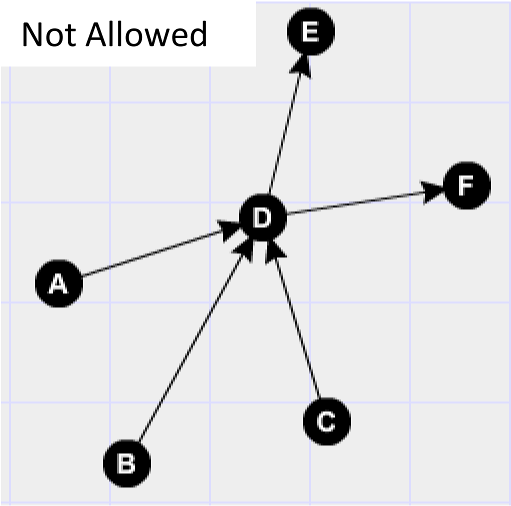



The main reasons behind such a constraint are:

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 reparing

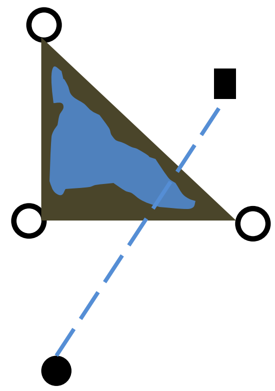
# Constraint 2: Power cables cannot be splitted




The out-degree of each turbine node must be one. However, in-degree can be more which is refered to as **branching** .

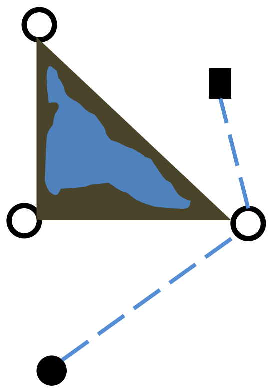
# Constraint 3: Restricted areas



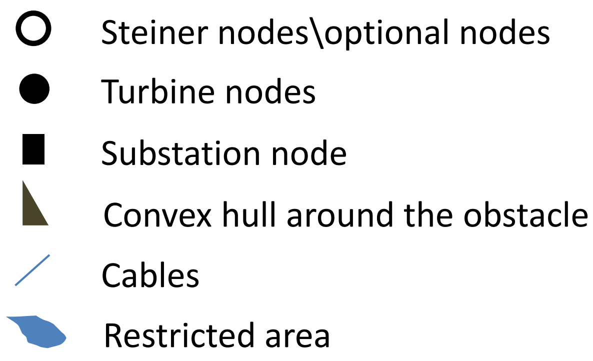
- Steiner nodes\optional nodes
- Turbine nodes
- Substation node
- Convex hull around the obstacle
- Cables
- Restricted area

- 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



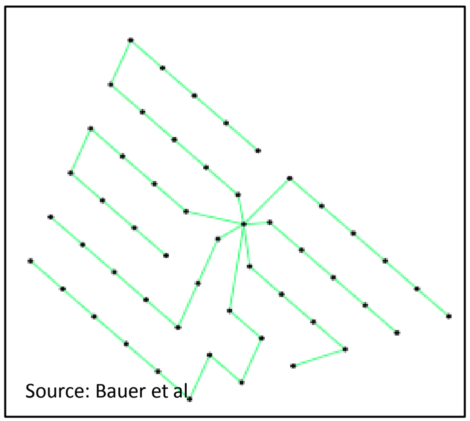
- Direct links are sometimes not possible due to restricted areas in the sea-bed
- Number of steiner nodes is a design parameter and can be more than the extreme points of the convex hull
- We are making an assumption that any concave and convex restricted area can be represented by a convex hull without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



# Allowed: Branching and parallel cables

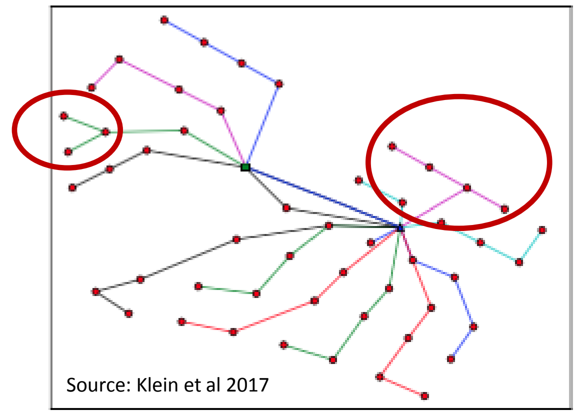
Both branching and parallel cables provide flexibility to the final layout and may lead to reduction in the total cable length

No-Branching



Source: Bauer et al

Branching



Source: Klein et al 2017

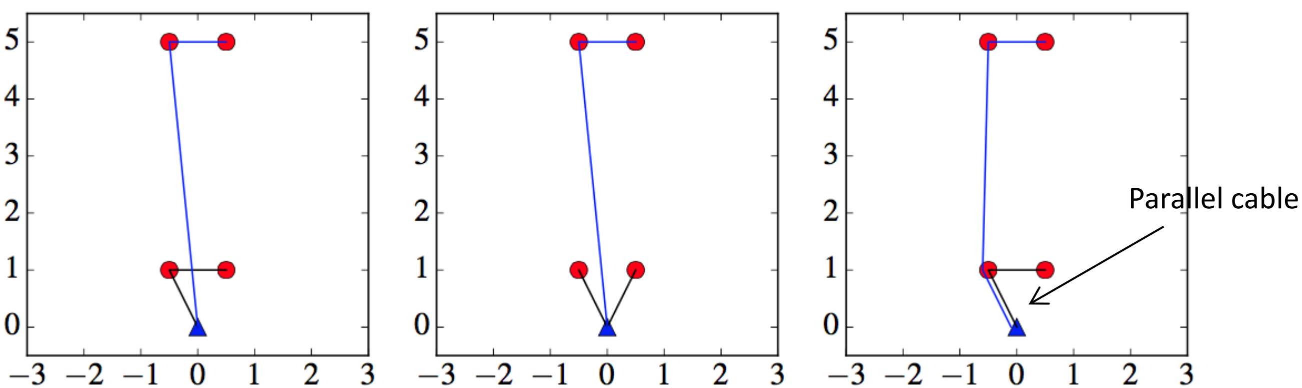



Image Source: Klein et al 2017

# 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

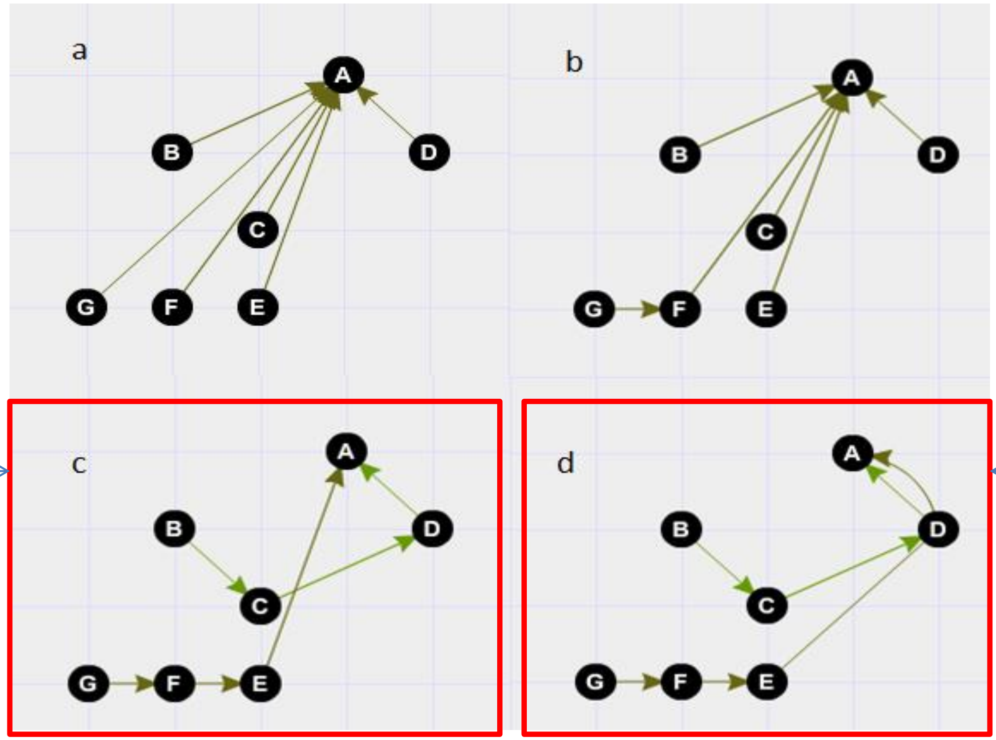
- Initial results from the Modified Esau-Williams (Wind farms: Walney 1 , Walney 2 and Barrow)
- Parametrization and introducing a shape factor
- Improved results
- From construction heuristic to Meta-Heuristic : Future activities (Very large Neighborhood search (VLNS) and GRASP)

# 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.

Final output of the Esau-Williams' Heuristic

Capacity = 3

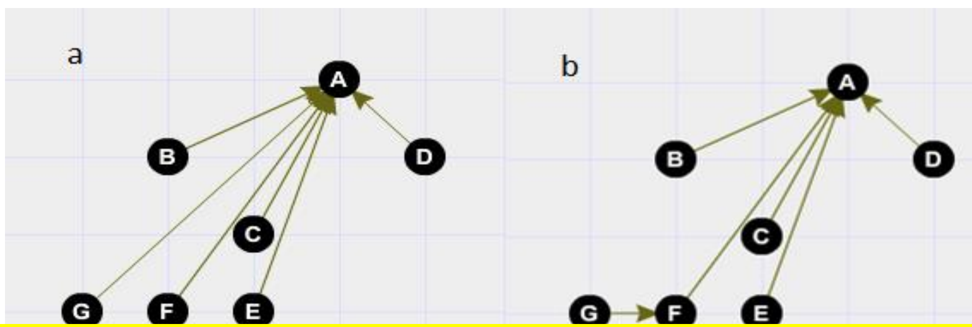



Feasible layout

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

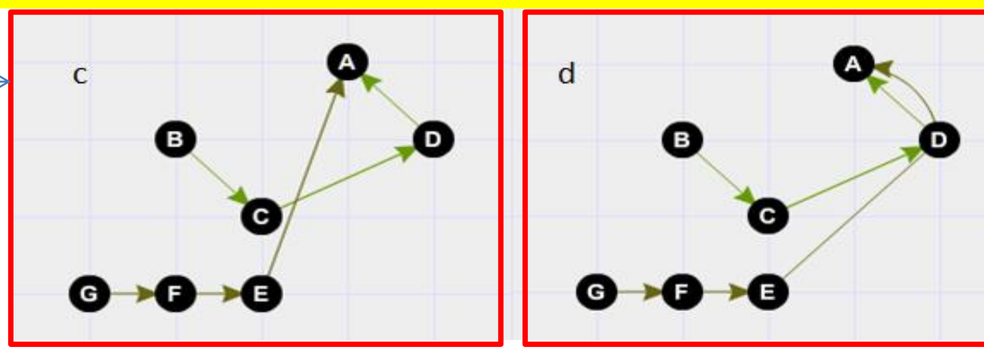
- **Esau-Williams'** heuristic is a well known heuristic for the capacitated minimum spanning tree problem.
- Start with a costly, feasible star layout
- In each iteration remove one link connecting the non-root node with the root node (substation node) resulting in cost saving.




Can't use Esau-Williams' heuristic alone!

Final output of the Esau-Williams' Heuristic

Capacity = 3




Feasible layout

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 $G$
$M \leftarrow \emptyset$
**for** *node* $i \in V \backslash 0$ **do**
  $M \leftarrow M \cup \{(0,i)\}$ ; `// set of links in spanning tree`
  $X_i \leftarrow \{i\}$
  $R_i = .1$ ; `// any value except 0`
**end**
**while** $(\exists i \in V : R_i > 0)$ **do**
  **for** *node* $i \in V$ **do**
    $j(i) \leftarrow$ nearest neighbouring node ; `// using (2.2.9)`
    compute $R_i$ ; `// Reduction value using (2.2.10)`
  **end**
  **find** $i^* \in \arg\max_{i \in V} R_i$
  **if** $R_{i^*} > 0$ **then**
    $M \leftarrow M \cup \{(j(i^*), i^*)\}$
    $M \leftarrow M \backslash \{(0, i^*)\}$
    $X_{j(i^*)} \leftarrow X_{i^*} \leftarrow X_{i^*} \cup X_{j(i^*)}$
  **end**
**end**
**return** $M$

V: set of vertices
A: $A \subseteq (V \times (V \backslash \{0\}))$
0: root node
c: cost of the arcs
K: cable capacity
$R_i$: reduction fuction value of node i
$X_i$: connected component containing node i

$$S(i) = \{j \in V \backslash 0 : j \notin X_i, (j,i) \in A, |X_i| + |X_j| \le K\} \qquad (2.2.8)$$

$$j(i) = \begin{cases} \text{one of the } j \in \arg\min_{j \in V \backslash \{0\}} \{c_{ji} : j \in S(i)\}, & S(i) \neq \emptyset \\ 0, & S(i) \neq \emptyset \end{cases} \qquad (2.2.9)$$

$$R_i = \begin{cases} c_{0i} - \min\{c_{ji} : j \in S(i)\}, & S(i) \neq \emptyset \\ 0, & S(i) \neq \emptyset \end{cases} \qquad (2.2.10)$$

# Pseudocode of Esau-Williams' heuristic

**Algorithm 1** Esau Williams

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

**Result:** $M$: set of $|V|-1$ arcs spanning $G$

$M \leftarrow \emptyset$

**for** *node* $i \in V \backslash 0$ **do**

  $M \leftarrow M \cup \{(0, i)\}$ ; // set of links in spanning tree

  $X_i \leftarrow \{i\}$

  $R_i = .1$ ; // any value except 0

**end**

**while** $(\exists i \in V : R_i > 0)$ **do**

  **for** *node* $i \in V$ **do**

    $j(i) \leftarrow$ nearest neighbouring node ; // using (2.2.9)

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

  **end**

  **find** $i^* \in \arg\max_{i \in V} R_i$

  **if** $R_{i^*} > 0$ **then**

    $M \leftarrow M \cup \{(j(i^*), i^*)\}$

    $M \leftarrow M \backslash \{(0, i^*)\}$

    $X_{j(i^*)} \leftarrow X_{i^*} \leftarrow X_{i^*} \cup X_{j(i^*)}$

  **end**

**end**

**return** $M$

V: set of vertices

A: $A \subseteq (V \times (V \backslash \{0\}))$

0: root node

c: cost of the arcs

K: cable capacity

$R_i$: reduction fuction value of node i

$X_i$: connected component containing node i

$$S(i) = \{j \in V \backslash 0 : j \notin X_i, (j,i) \in A, |X_i| + |X_j| \le K\} \tag{2.2.8}$$

$$j(i) = \begin{cases} \text{one of the } j \in \arg\min_{j \in V \backslash \{0\}} \{c_{ji} : j \in S(i)\}, & S(i) \neq \emptyset \\ 0, & S(i) \neq \emptyset \end{cases} \tag{2.2.9}$$

$$R_i = \begin{cases} c_{0i} - \min\{c_{ji} : j \in S(i)\}, & S(i) \neq \emptyset \\ 0, & S(i) \neq \emptyset \end{cases} \tag{2.2.10}$$

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)

# Idea to tackle cable crossing

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

```
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_k i_{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 ← ∅, M ← ∅
while any(R) do
    /* #1                                                   */
    FirstTime ← 0
    while FirstTime≠ 1 do
        /* #2                                               */
        for node i ∈ T do
            compute R_i ;            // 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)

**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 ← ∅,  M ← ∅
while any(R) do
    /* #1                                                      */
    FirstTime ← 0
    while FirstTime≠ 1 do
        /* #2                                                  */
        for node i ∈ T do
            compute R_i ;          // 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

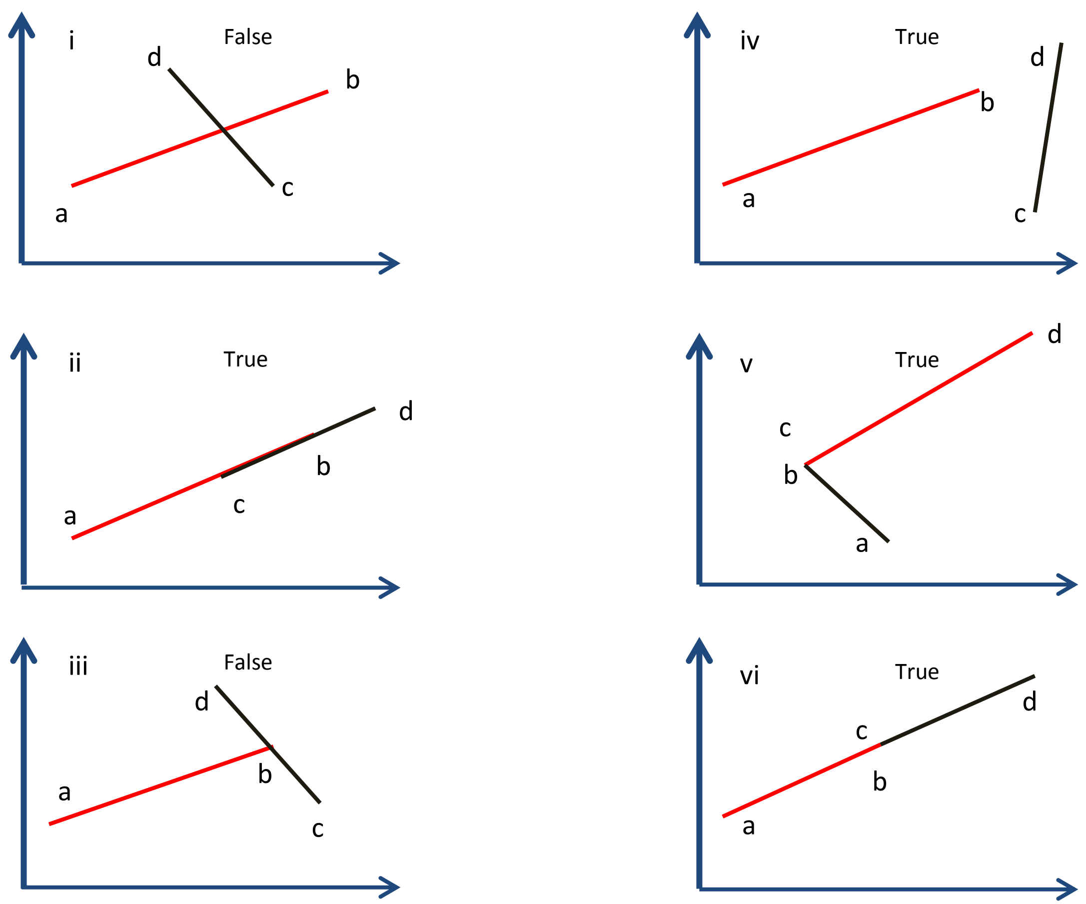
We have selected node $\mathbf{i_0}$ having the maximum reduction function value and its nearest node $\mathbf{i_n}$ . Now, in pre-processing stage the shortest feasible path between them is searched which may or may not be a direct arc

# Obstacle-Aware Esau Williams Heuristic(2/2)

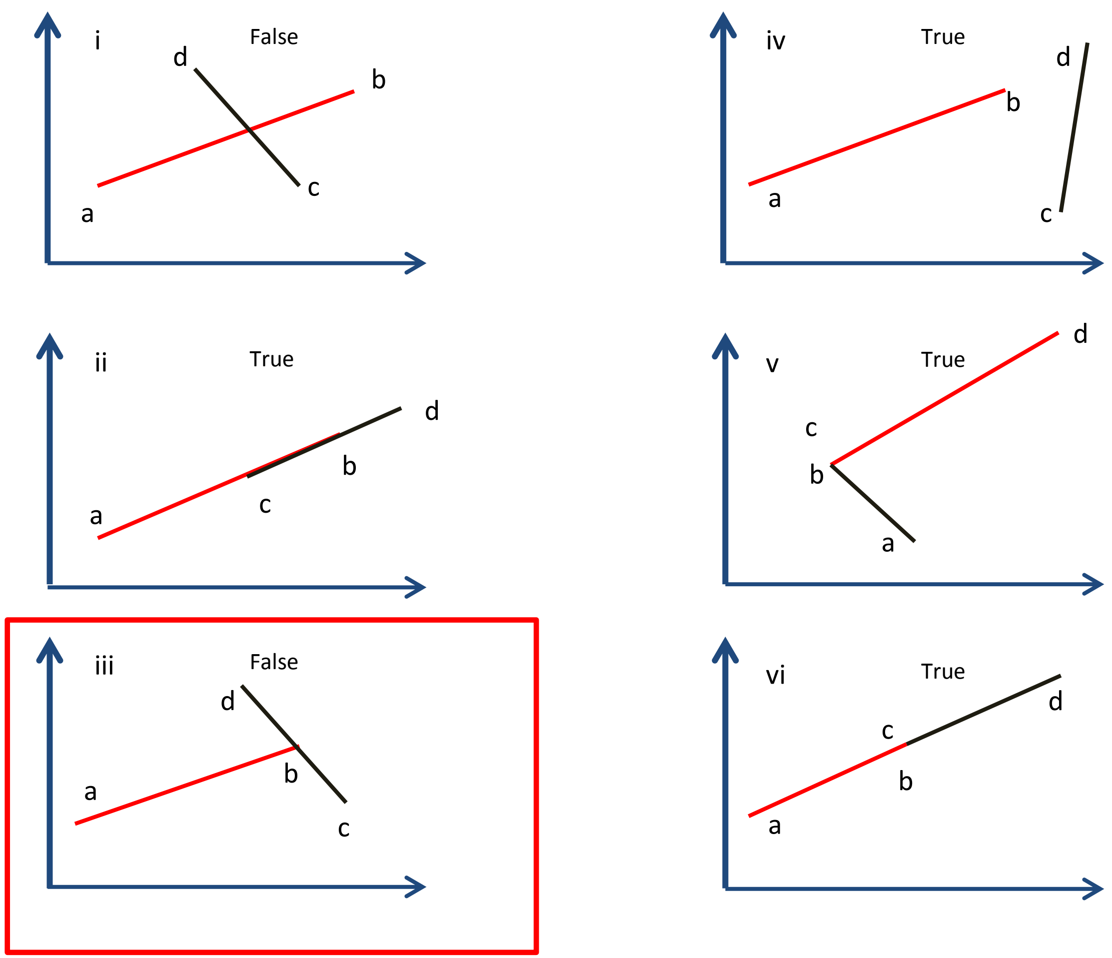
```
        /* #3                                                        */
    while crossingSwitch == TRUE do
        FirstTime ← FirstTime + 1
            crossingSwitch ← FALSE
            DJ ← Dijkstra(G, i_0, i_n)   /* shortest path between
            i_0 and i_n                                              */
        for each arc (i_k, i_{k+1}) in the shortest path DJ do
            if Non-Crossing(IntersectionArray, i_k, i_{k+1}) == FALSE
             then
                c_{i_k i_{k+1}} ← ∞;                  // any high value
                crossingSwitch ← TRUE
            end
        end
        if !crossingSwitch&&FirstTime ≠ 1 then
            C[i_0][i_n] ← c_{i_0 i_1} + c_{i_2 i_3} + ··· + c_{i_{n-1} i_n};   // exit loop #3
        end
    end
  end
  post node joining steps and updates of connected components, L2, M
end
forming shortest feasible paths from an active node in each connected
 component to the substation
 return M
```

Pre-processing stage

# Non-Crossing procedure’s output

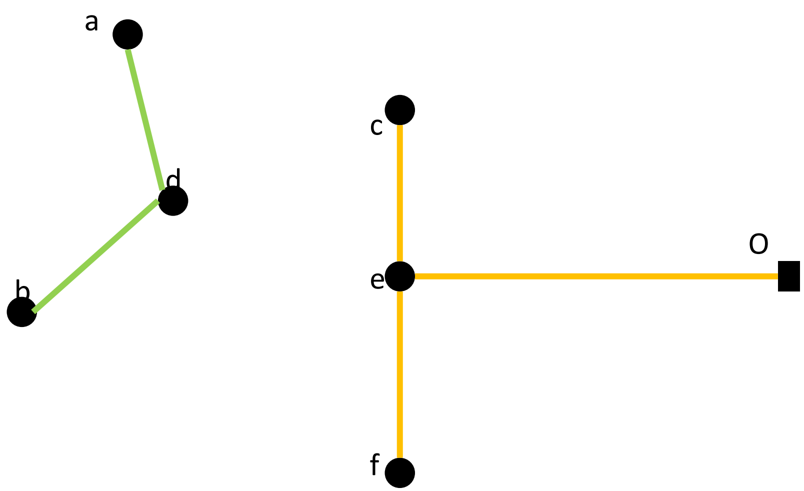


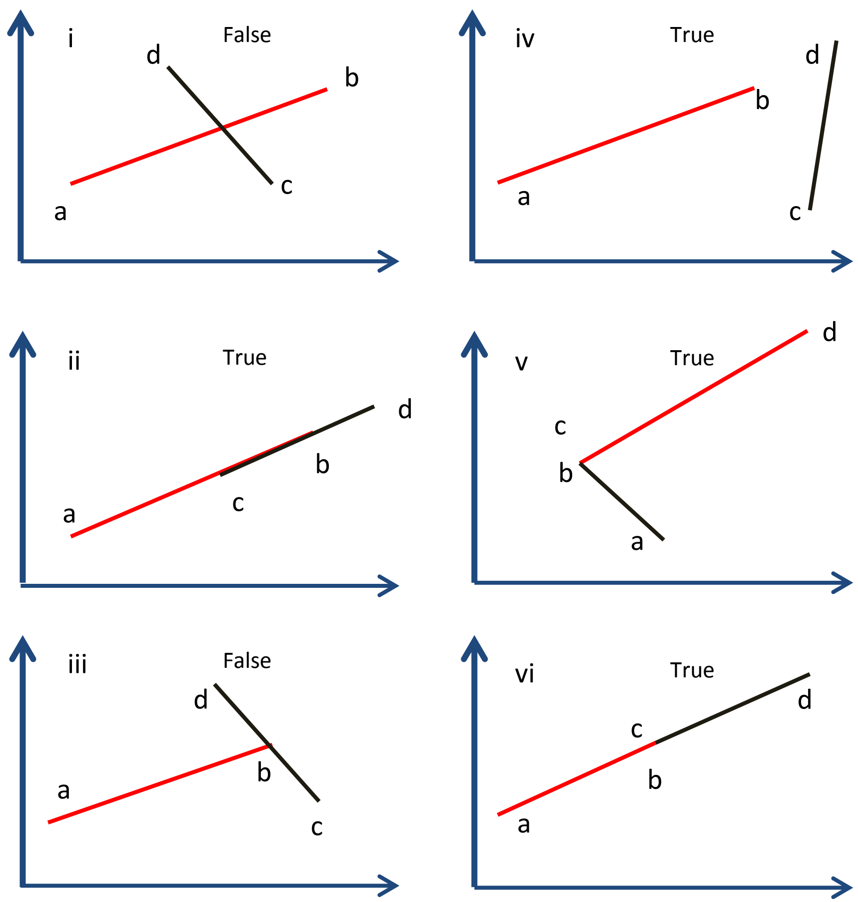
# Non-Crossing procedure's output



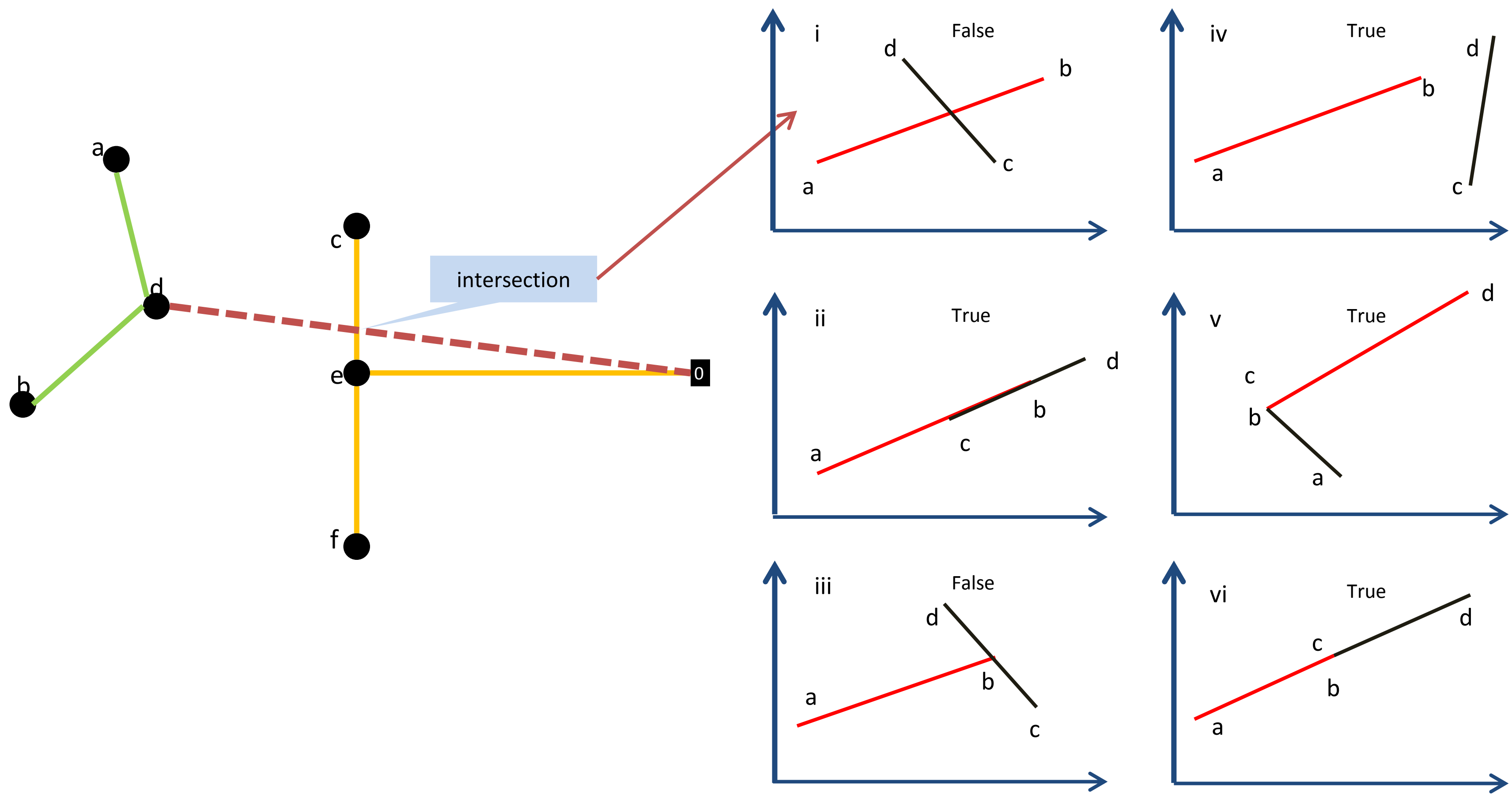

Note: Non-crossing procedure always assesses pair of line segments (one from IntersectionArray and one of the edge in the shortest path given by Dijkstra)

# Challange: Non-Crossing procedure is unable to identify node crossing

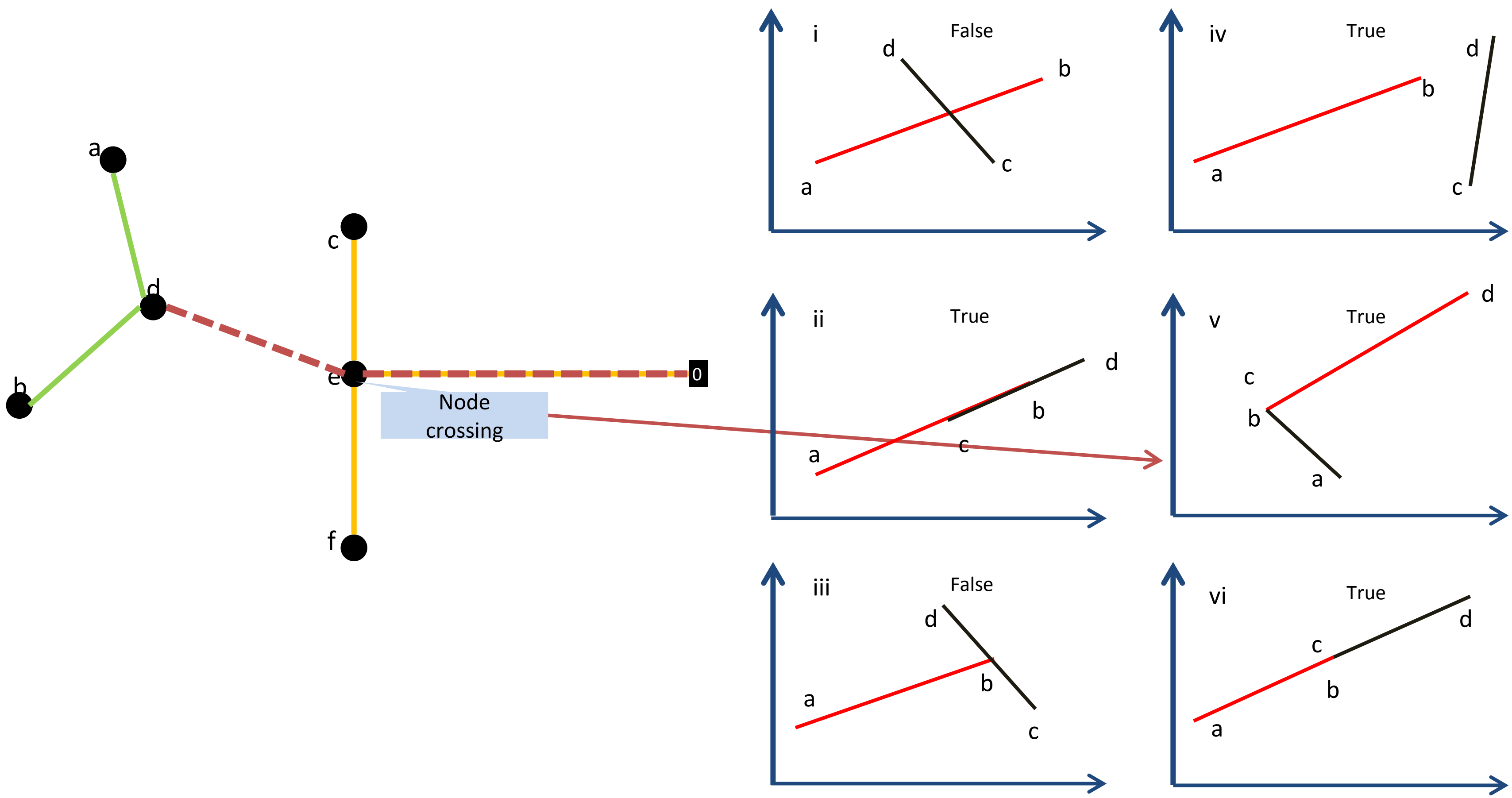


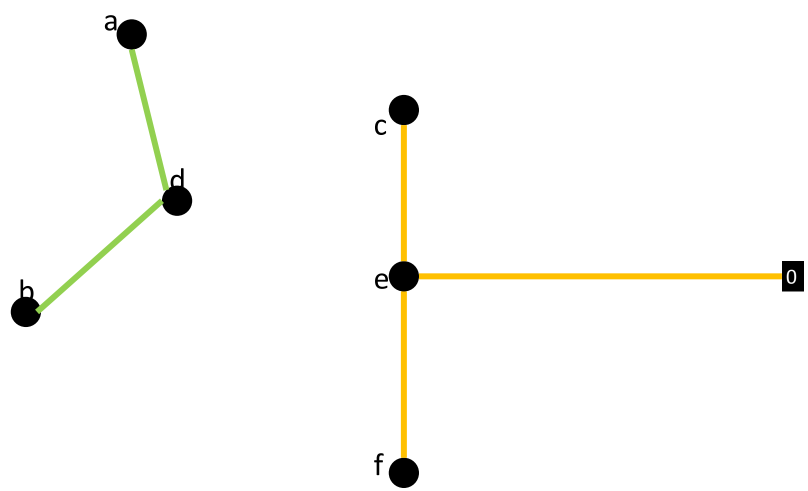
# Challange: Non-Crossing procedure is unable to identify node crossing

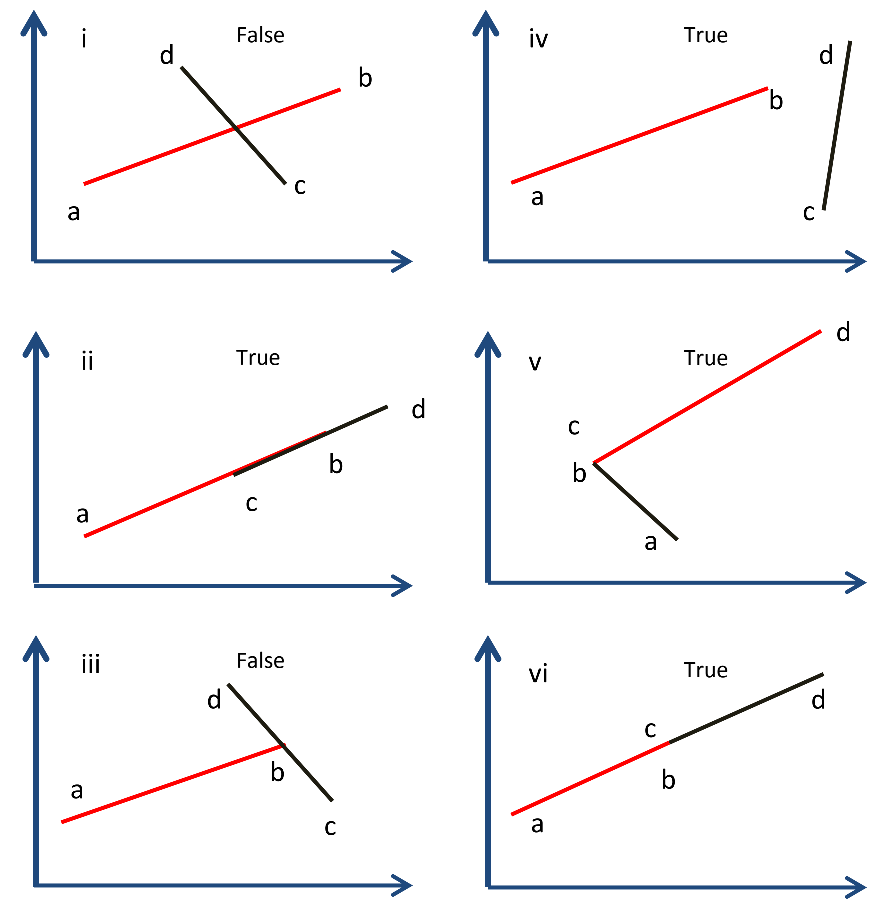


# Challange: Non-Crossing procedure is unable to identify node crossing

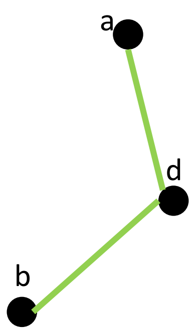


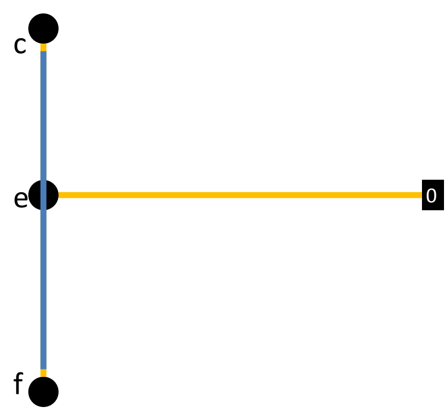
# Challange: Non-Crossing procedure is unable to identify node crossing

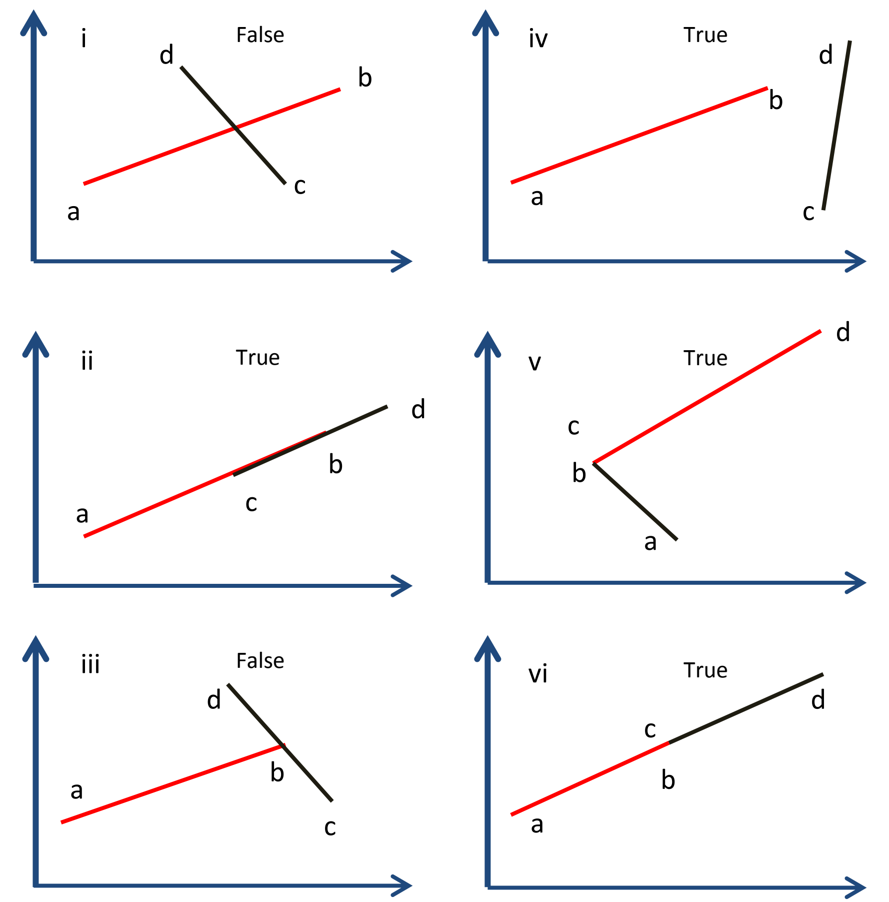


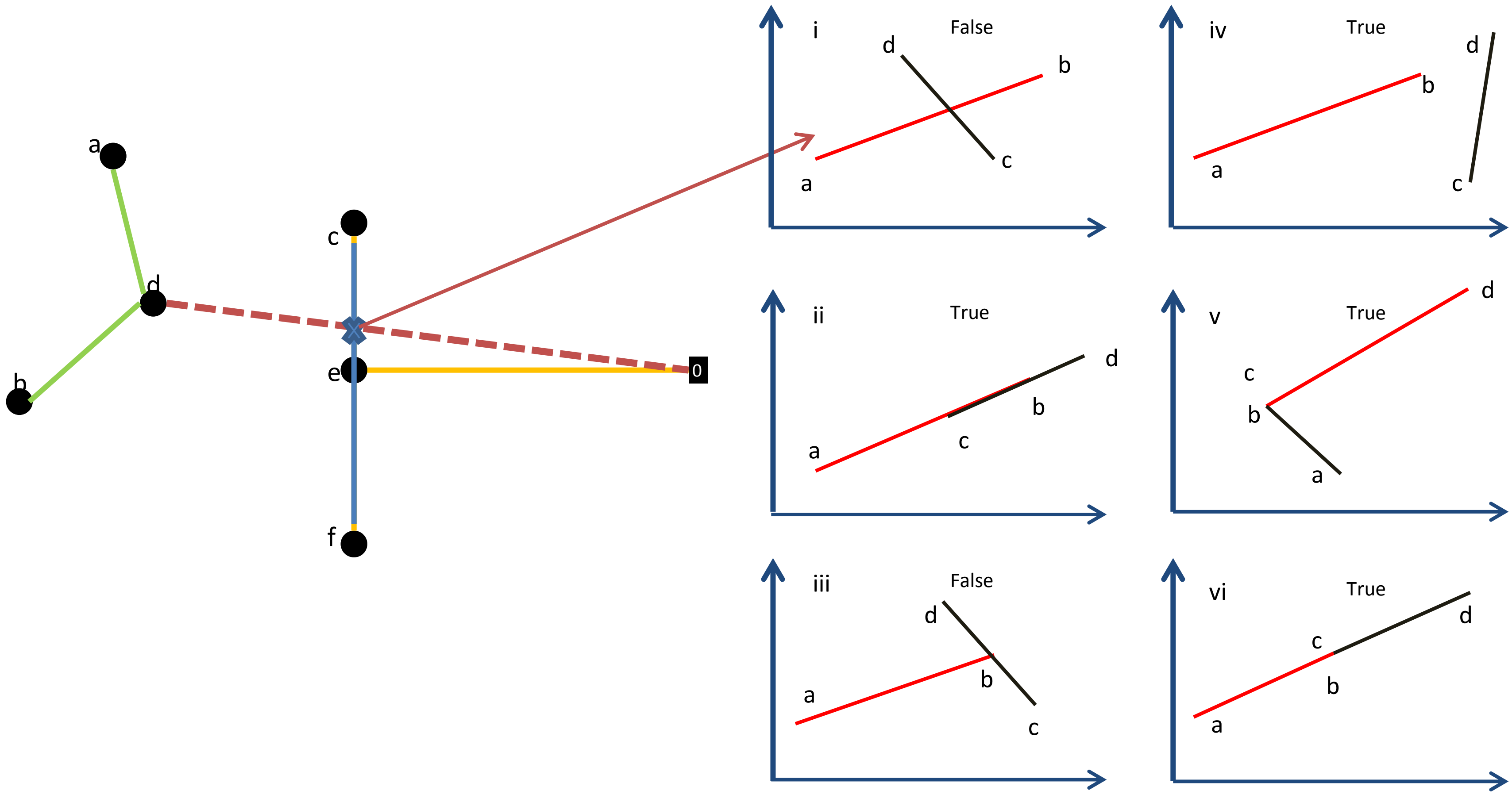
# Solution(1/4): Add new line segments such that node crossings are detected by Non-crossing procedure



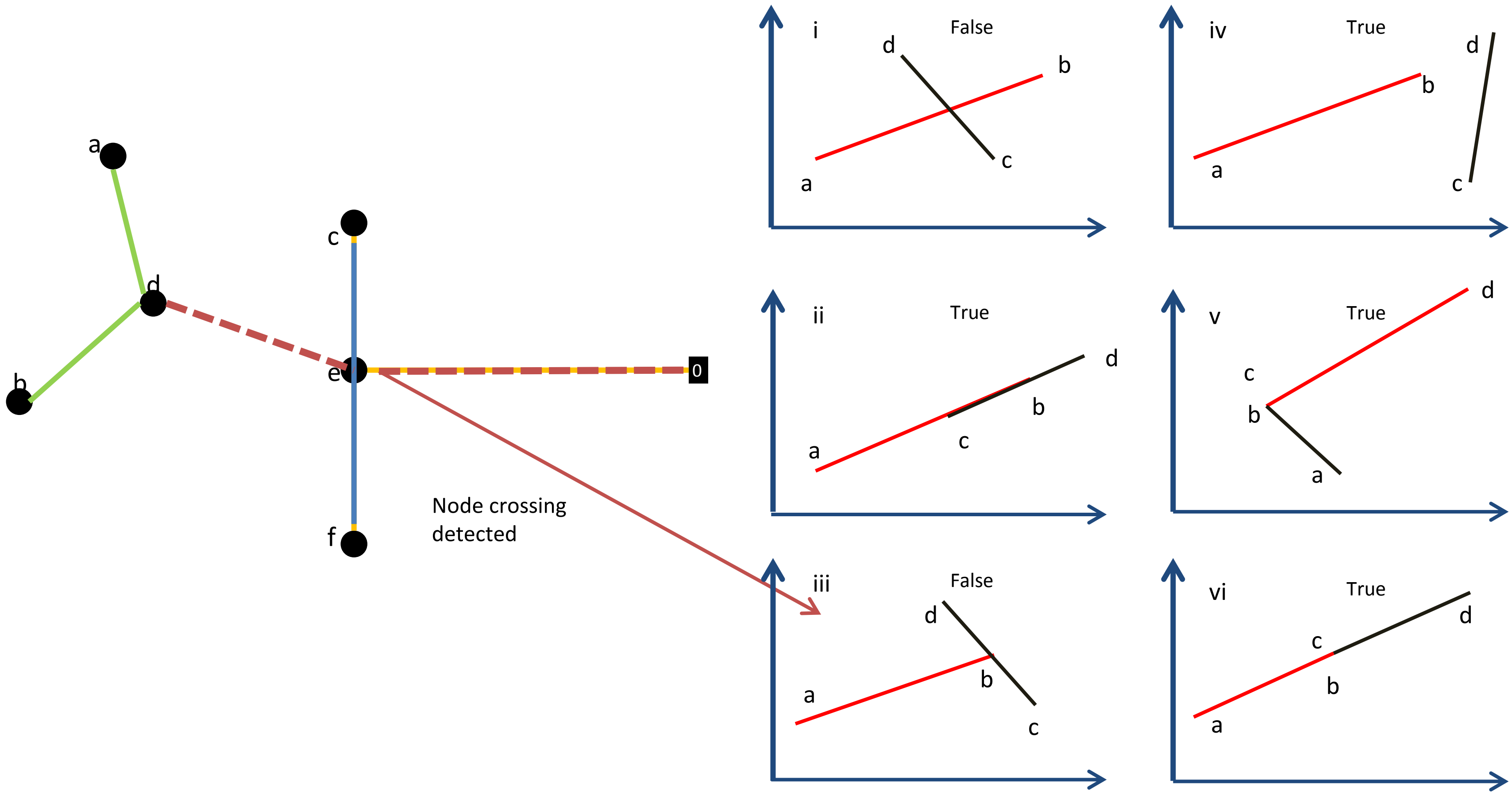



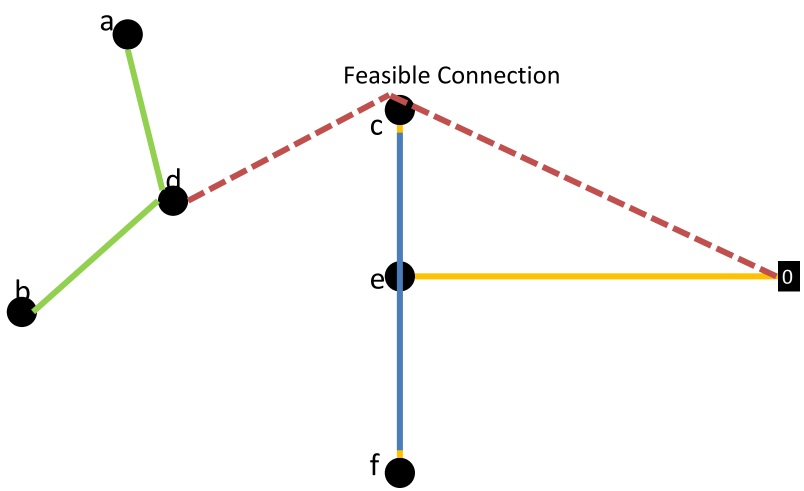
# Solution(2/4): Add new line segments such that node crossings are detected by Non-crossing procedure

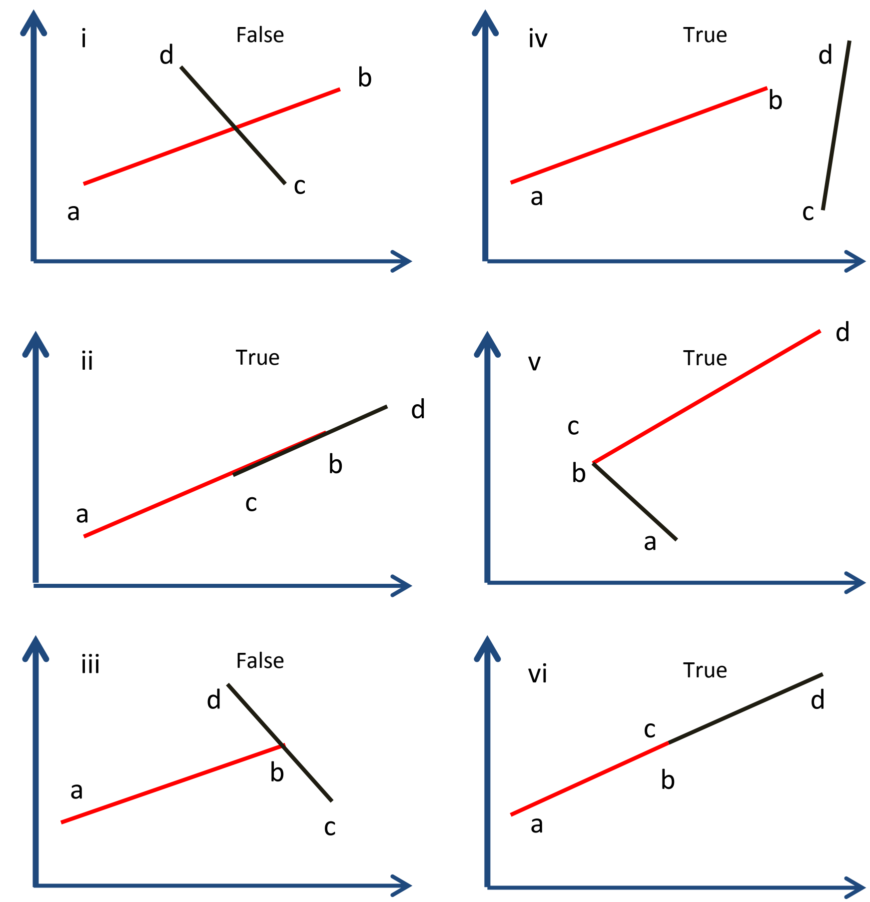


# Solution(3/4): Add new line segments such that node crossings are detected by Non-crossing procedure



# Solution(4/4): Add new line segments such that node crossings are detected by Non-crossing procedure



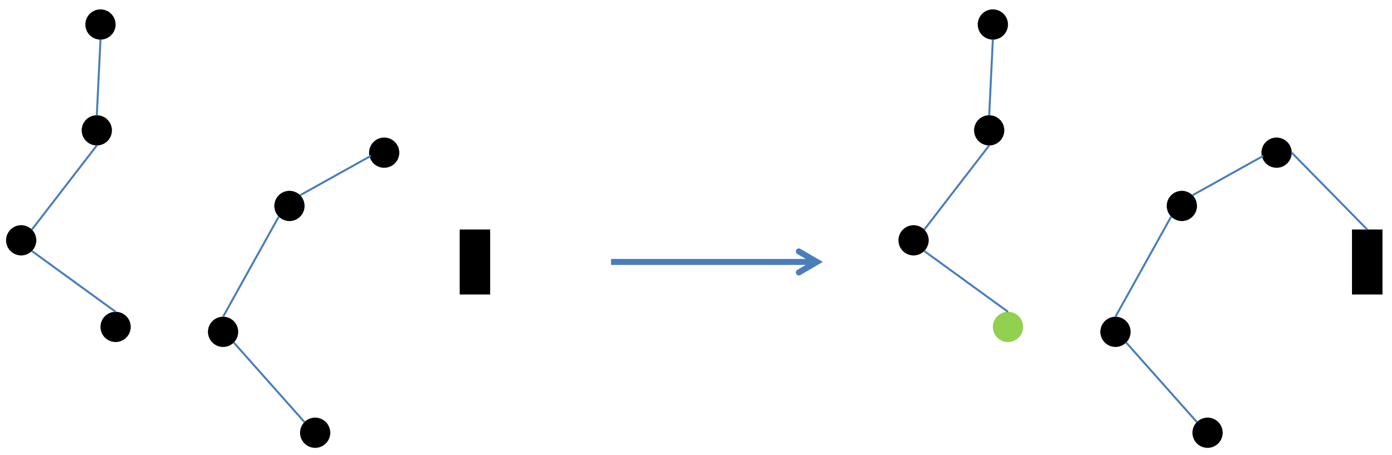



# Solution(1/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post partitioning step

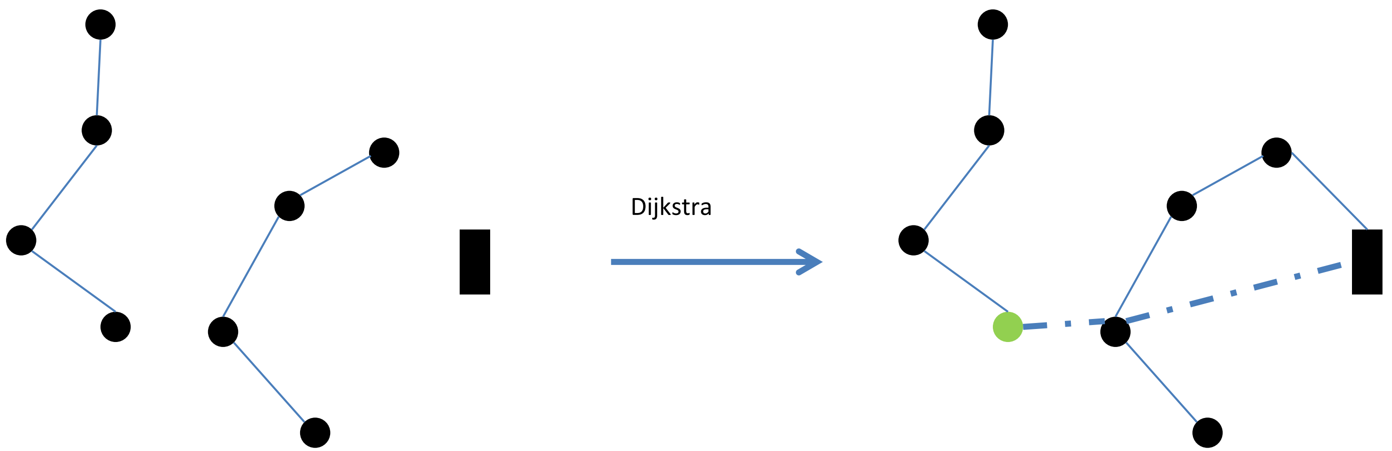


# Solution(1/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post partitioning step

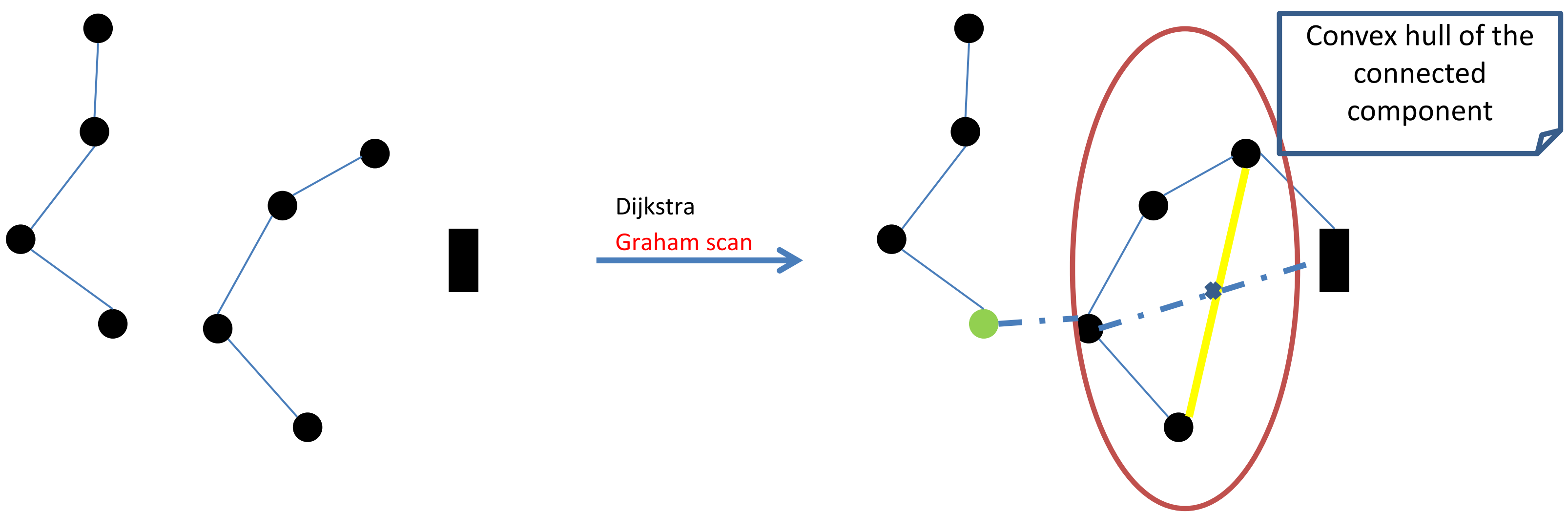


# Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post joining step

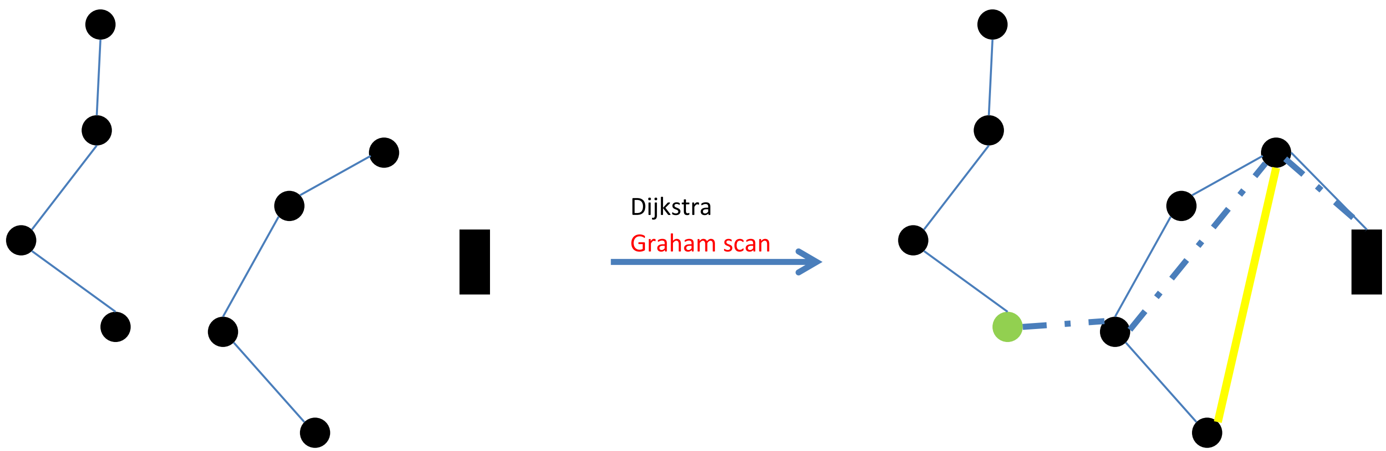


# Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components ← Output from 1st part of the algorithm

↓

**Post joining step**

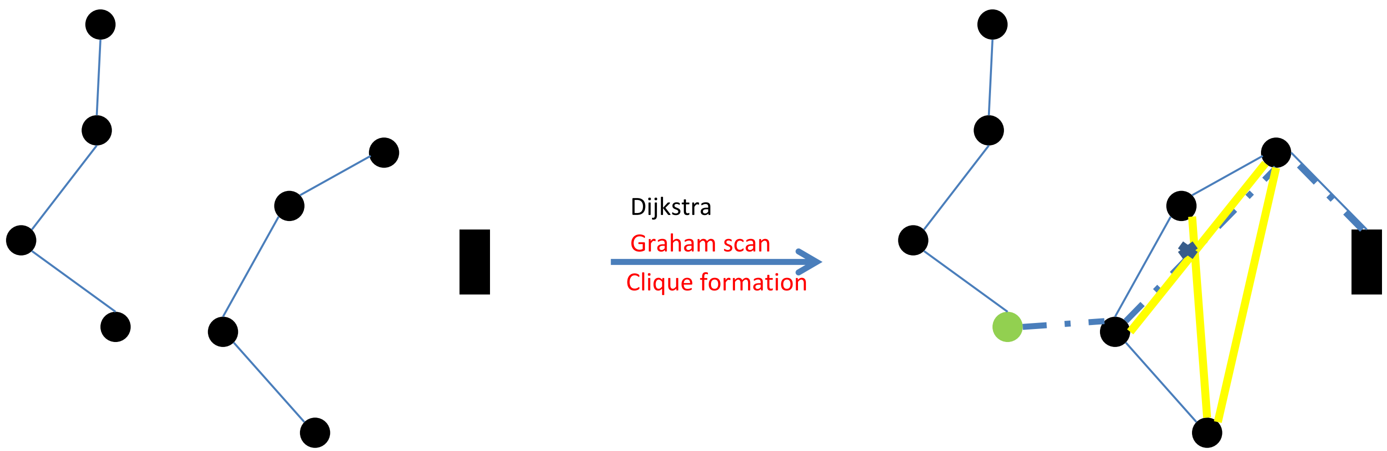


# Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components

Output from 1st part of the algorithm

Post joining step

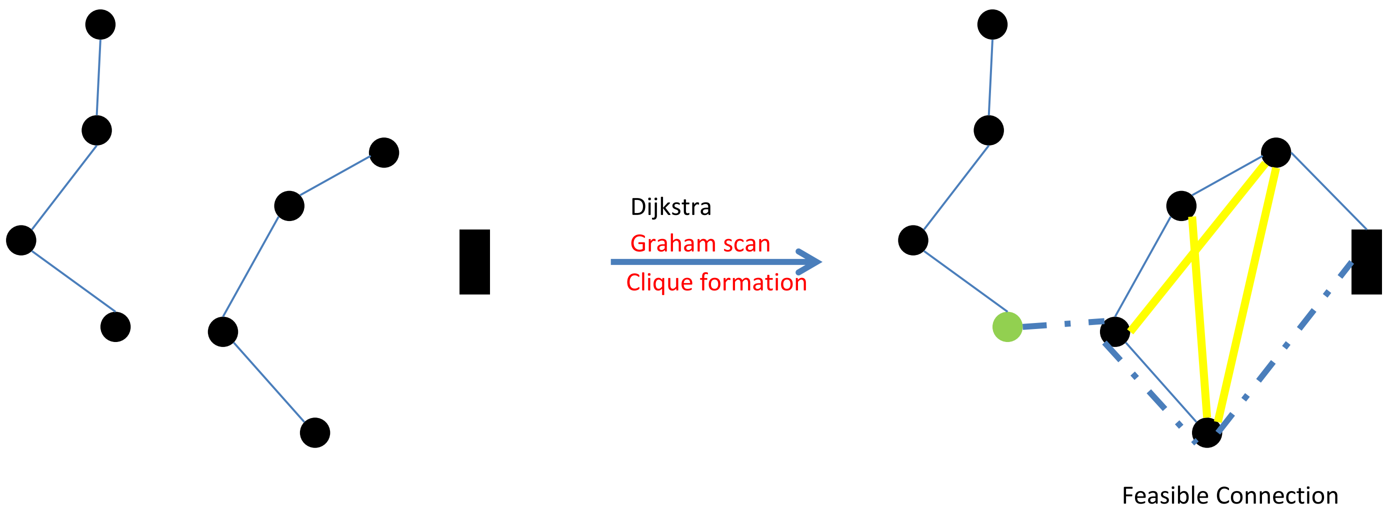


# Solution(2/2): Where to add the line segments?

Partitioning of turbine nodes in different connected components ← Output from 1st part of the algorithm

↓

**Post joining step**




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

# 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

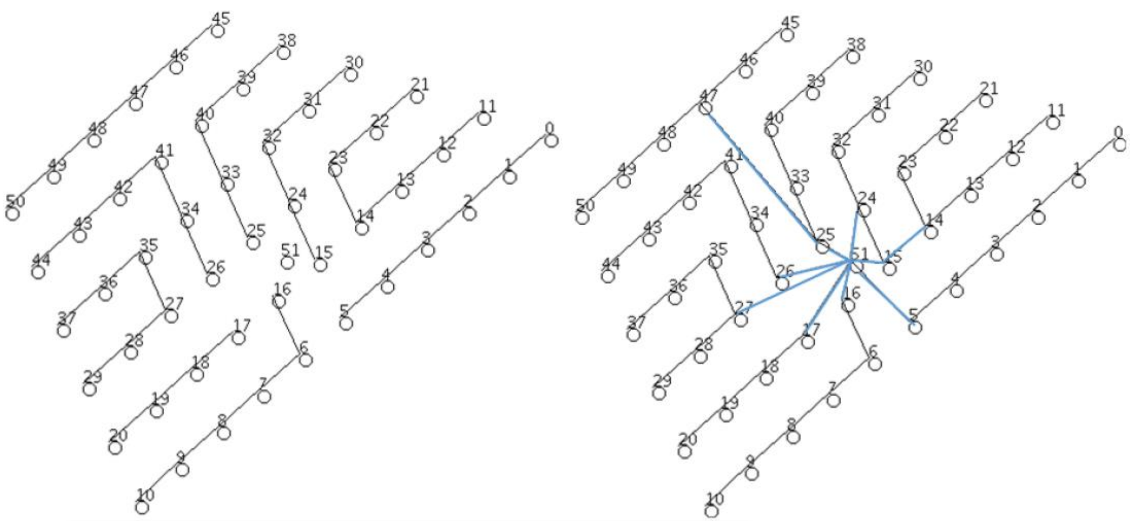
**Developed Heuristic :**

- 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 **[cost(heuristic)/cost(optimal solution)]**

# Experimental Results-2

| K | Barrow(T=30) | | | Walney-1(T=51) | | | Walney-2(T=51) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Exact | Alg 2 | gap | Exact | Alg 2 | gap | Exact | Alg 2 | gap |
| 2 | 36990 | 38115 | 1.03 | 70286 | 75105 | 1.07 | 97885 | 117849 | 1.20 |
| 4 | 23208 | 23243 | 1.00 | 47411 | 49534 | 1.04 | 63496 | 73374 | 1.15 |
| 5 | 20691 | 21815 | 1.05 | 43420 | 44444 | 1.02 | 56904 | 62739 | 1.10 |
| 6 | 18374 | 20980 | 1.14 | 41418 | 43858 | 1.05 | 52981 | 63568 | 1.19 |

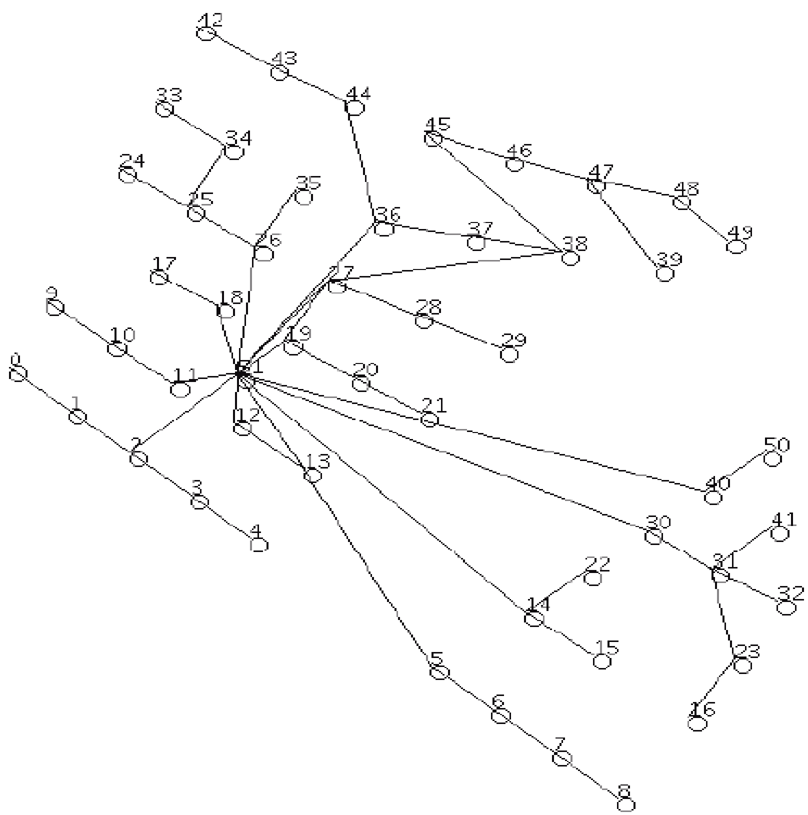
K= cable capacity , Alg2 = Obstacle-Aware Esau Williams



Walney 1 final layout for K=6

# Experimental Results-3

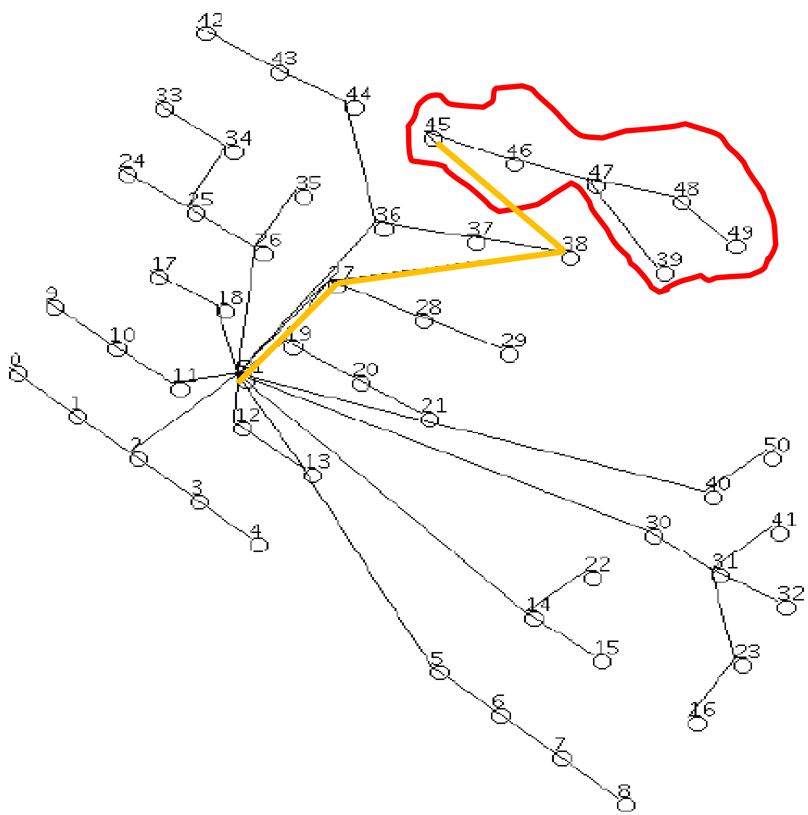
Walney 2 final layout for K=6




- 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

# Experimental Results-3

Walney 2 final layout for K=6




- ❑ 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

$$S(i) = \{j \in V \backslash 0 : j \notin X_i, (j,i) \in A, |X_i| + |X_j| \le K\} \qquad (2.2.11)$$

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

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

# Introducing weight parameter in reduction function

$$S(i) = \{j \in V \backslash 0 : j \notin X_i, (j,i) \in A, |X_i| + |X_j| \leq K\} \qquad (2.2.11)$$

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

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

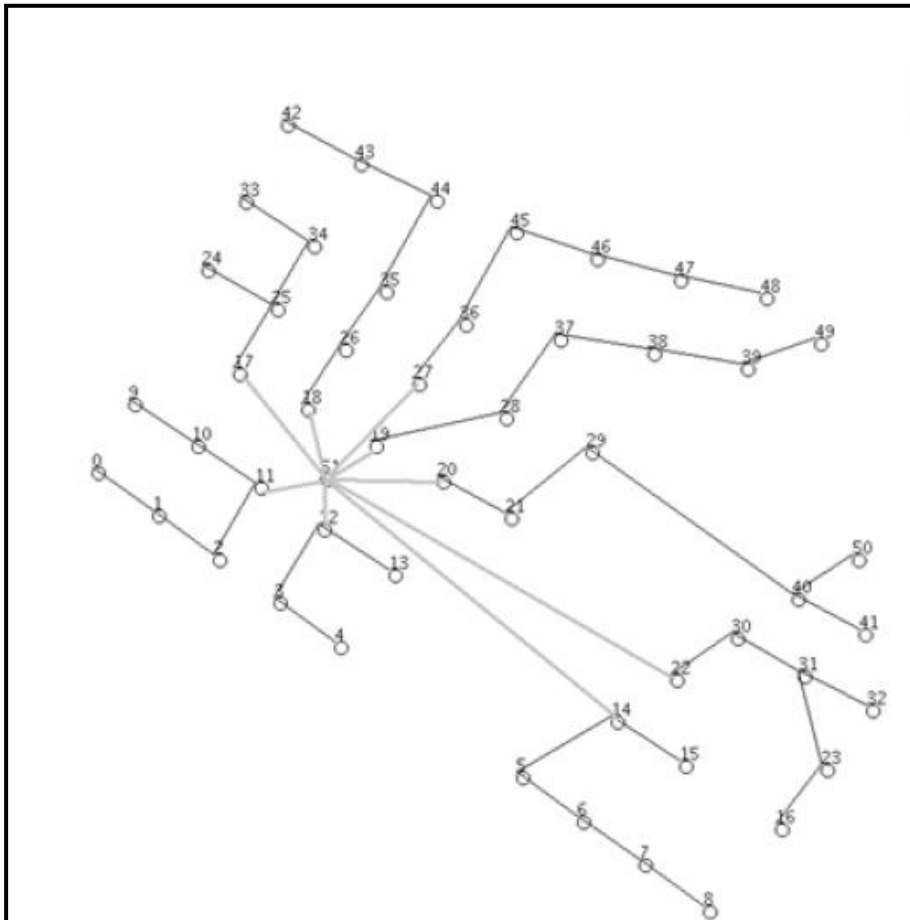
As the value of weight parameter W increases , turbine nodes closer to the substation will be preferred.

# Results from exact, obstacle aware Esau Williams and algorithm with weight parameter

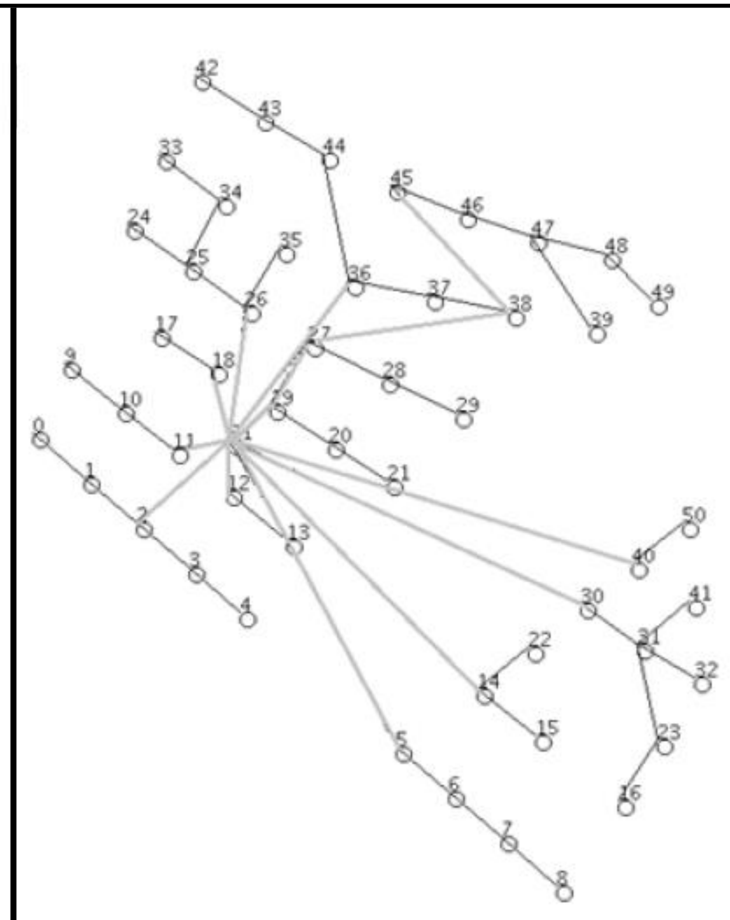
| Wind Farm | Exact | Obstacle-Aware | | Parametric | |
|---|---|---|---|---|---|
| K=6 | | value | gap | value | gap |
| Walney 1 | 41418 | 43858 | 1.05 | 42580 | 1.028 |
| Barrow | 18374 | 20980 | 1.14 | 18900 | 1.0286 |
| Walney 2 | 52981 | 63568 | 1.19 | 53214 | 1.004 |
| K=5 | | value | gap | value | gap |
| Walney 1 | 43420 | 44444 | 1.0235 | 43498 | 1.002 |
| Barrow | 20691 | 21815 | 1.054 | 21105 | 1.02 |
| Walney 2 | 56904 | 62739 | 1.1 | 57816 | 1.016 |
| K=4 | | value | gap | value | gap |
| Walney 1 | 47411 | 49534 | 1.044 | 48396 | 1.02 |
| Barrow | 232208 | 23243 | 1.001 | 23243 | 1.001 |
| Walney 2 | 63496 | 73374 | 1.15 | 63579 | 1.001 |

# Improved result for Walney 2

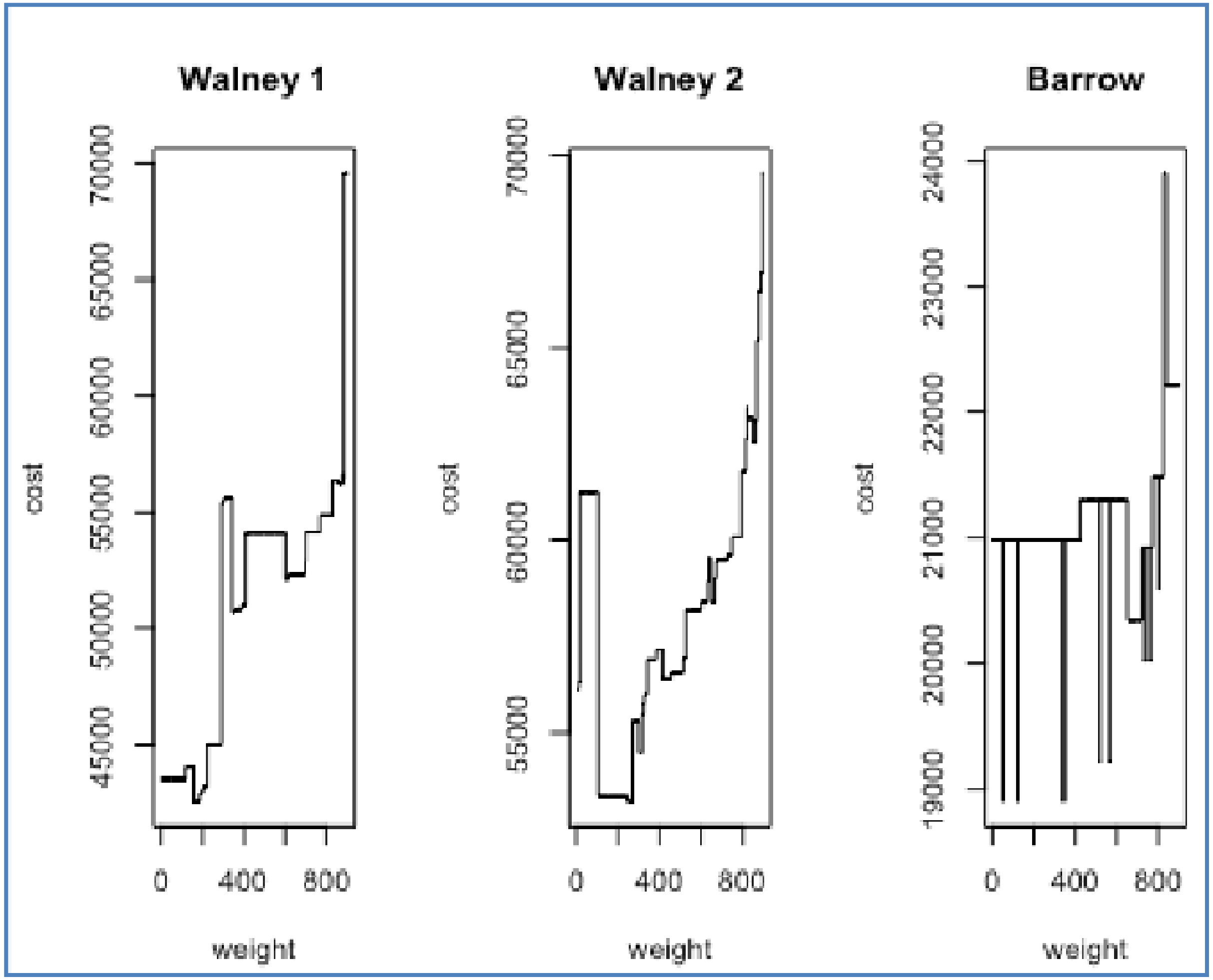
Walney 2 (Modified Esau Williams Algorithm-Parametric)



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.*