Universidad
Politécnica
de Cartagena

Universidad Politécnica de Cartagena

JJA ALGORITHM

AUTORES:
Jairo Peña Iglesias
José A. López Pastor

JJA ALGORITHM

Jairo Peña Iglesias	Jeffrey
José A. López Pastor	Jacob
ALGORITHM	Abrams

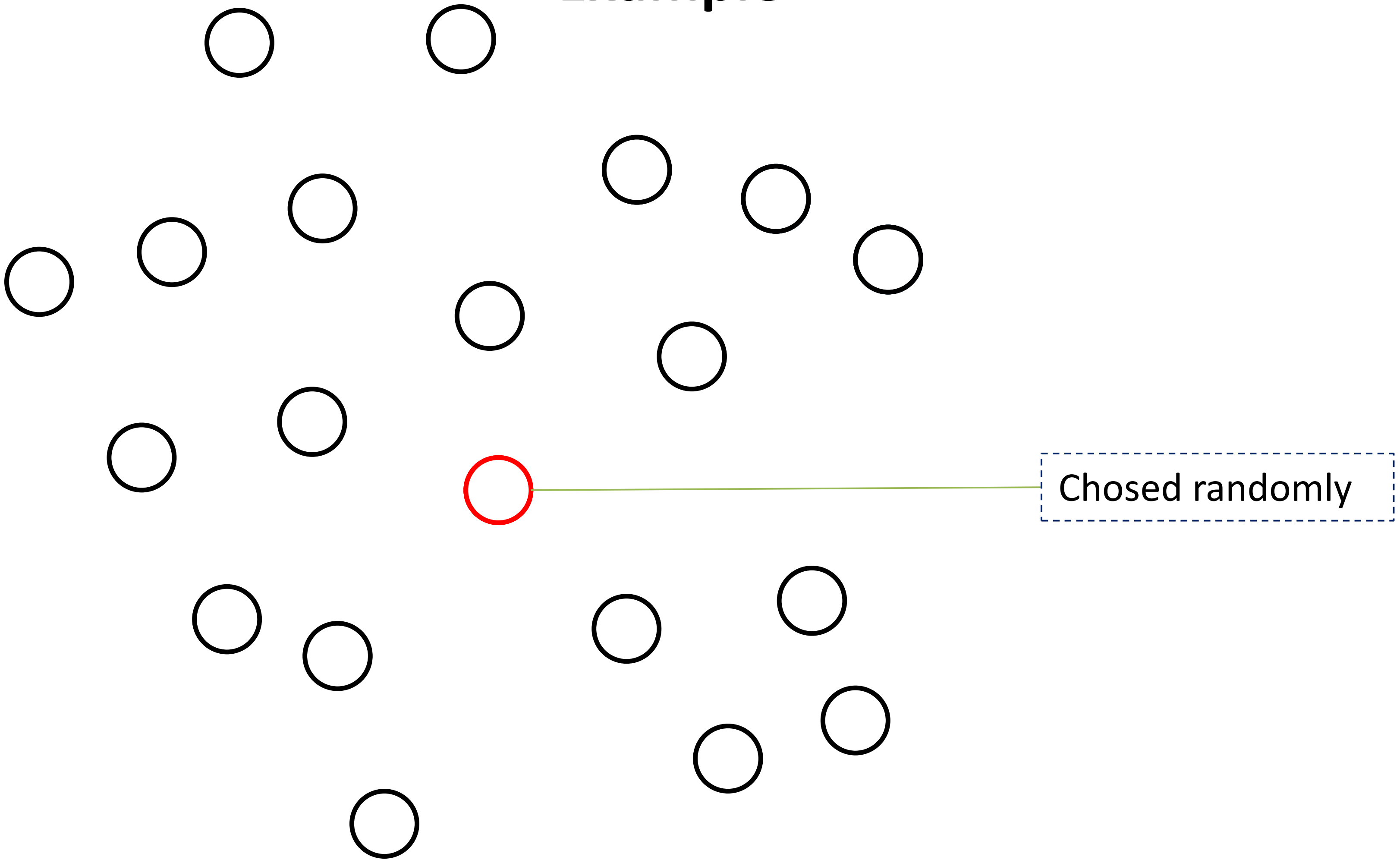
Initial considerations

Algorithm for an unique problem with specific parameters.

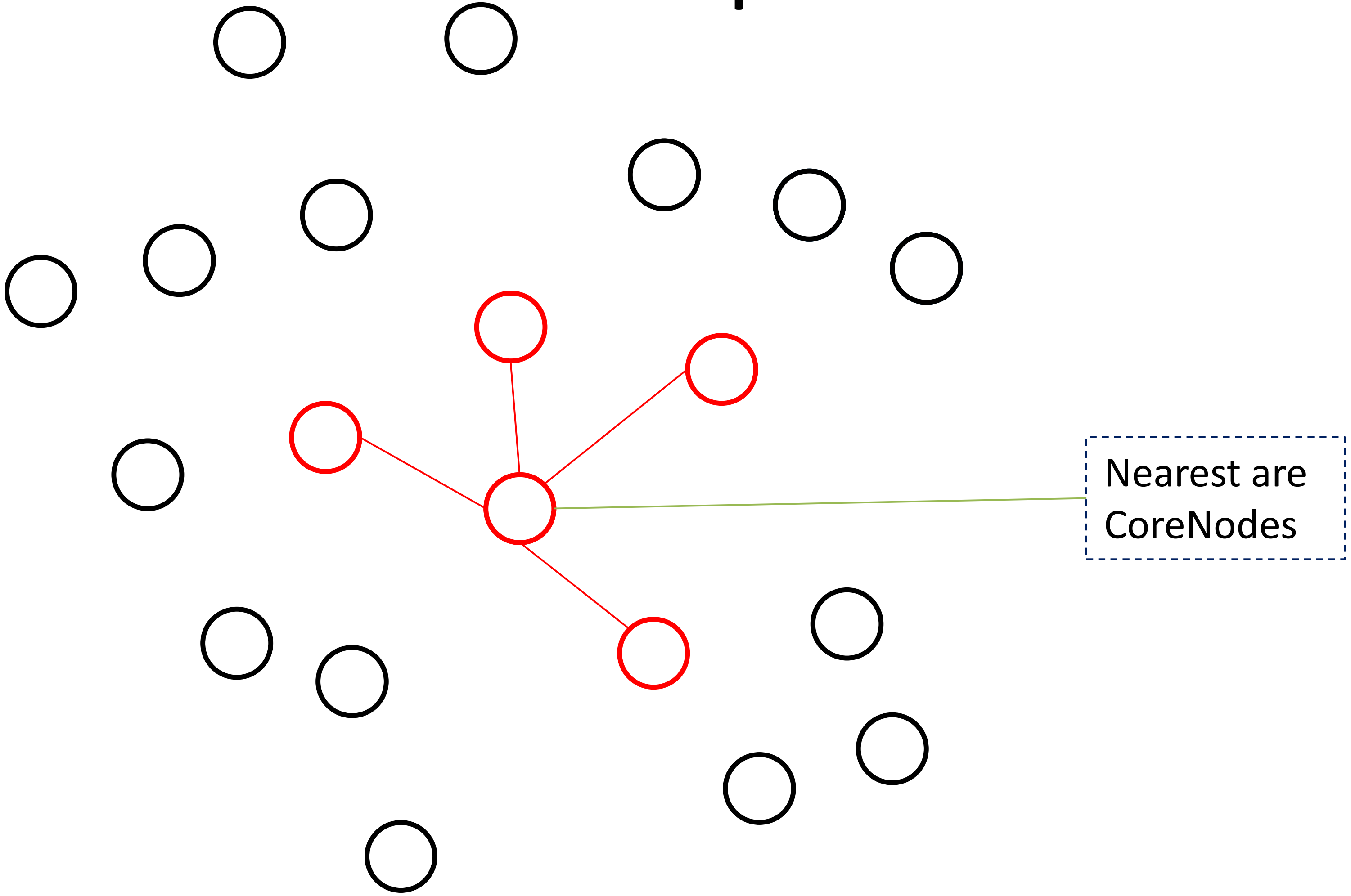
After some experimental calculations, we realized that:

- The cost of links between CoreNodes have a higher cost in the solution than rest of parameters. This is because CoreNodes links between them are more than CoreToAccess links.
- So as, our solution tries to **minimizate the cost of chose the CoreNodes and their links**.
- Thus, we have based on the Greedy Algorithm.

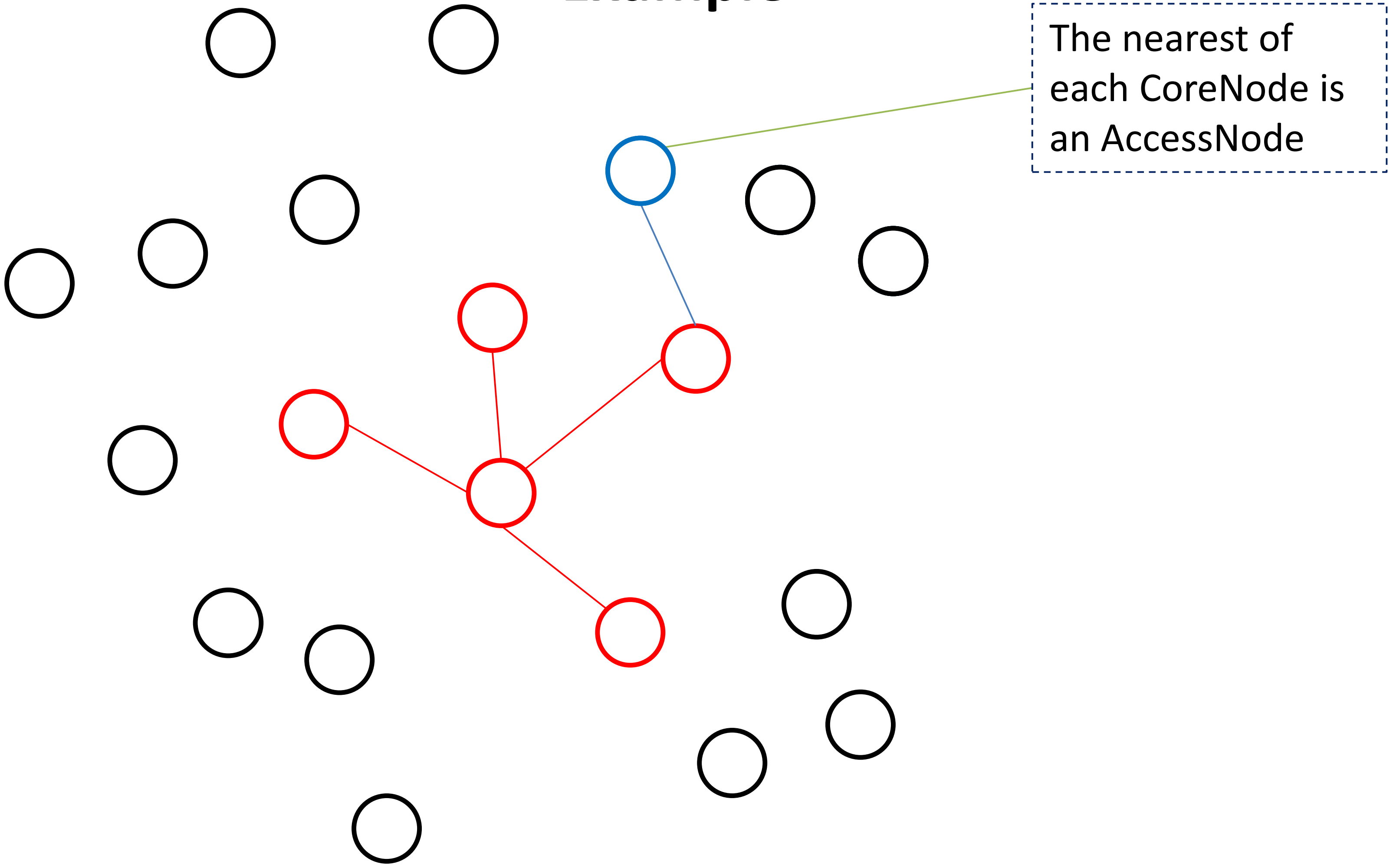
Example



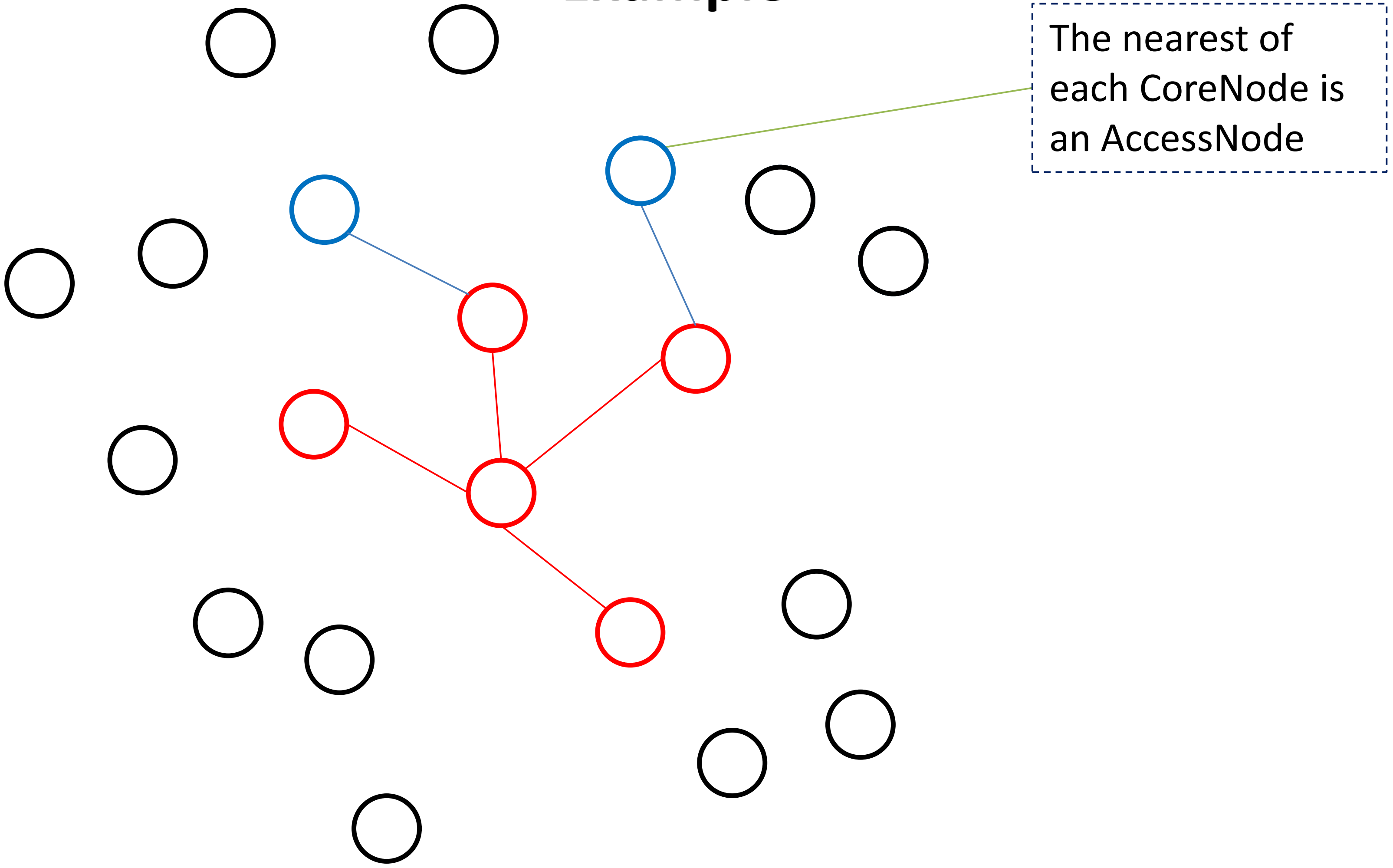
Example



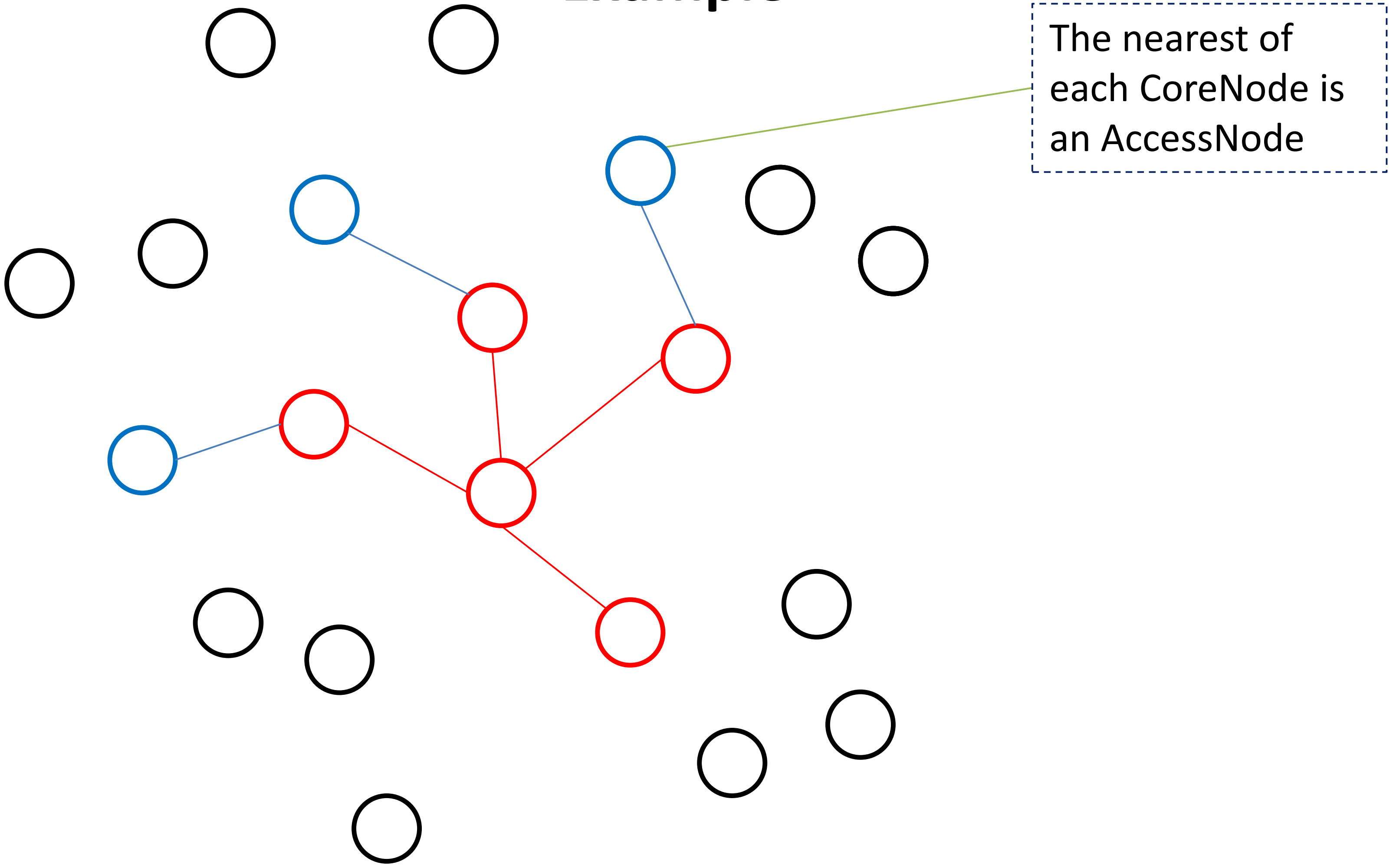
Example



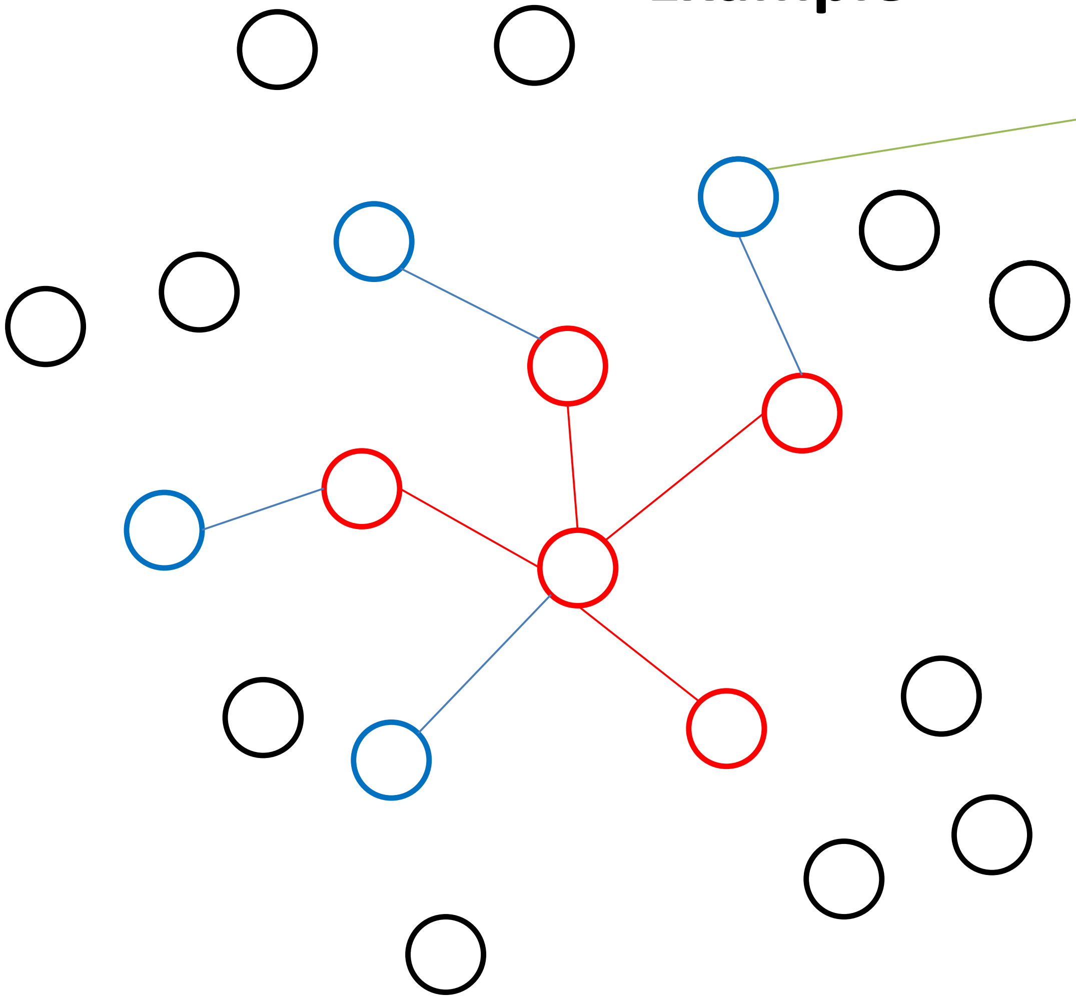
Example



Example

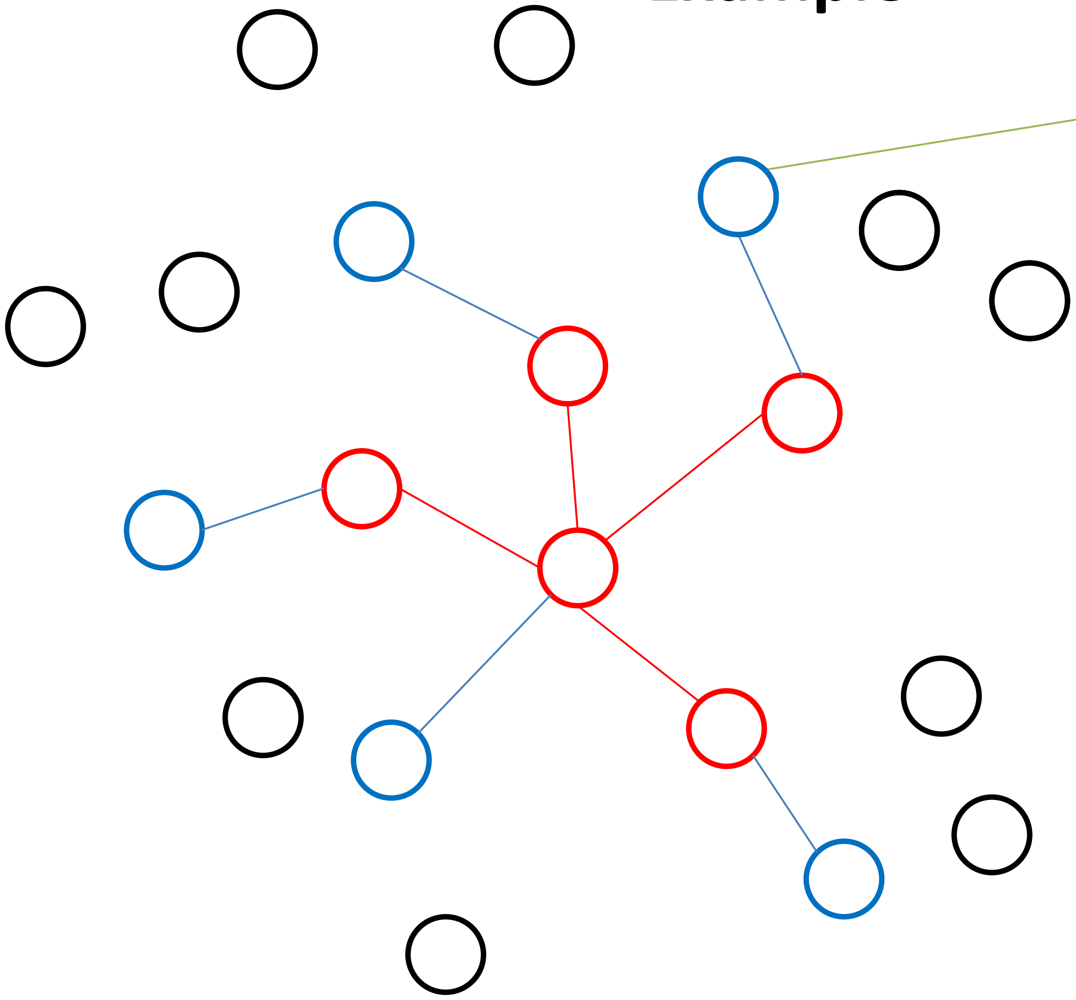


Example



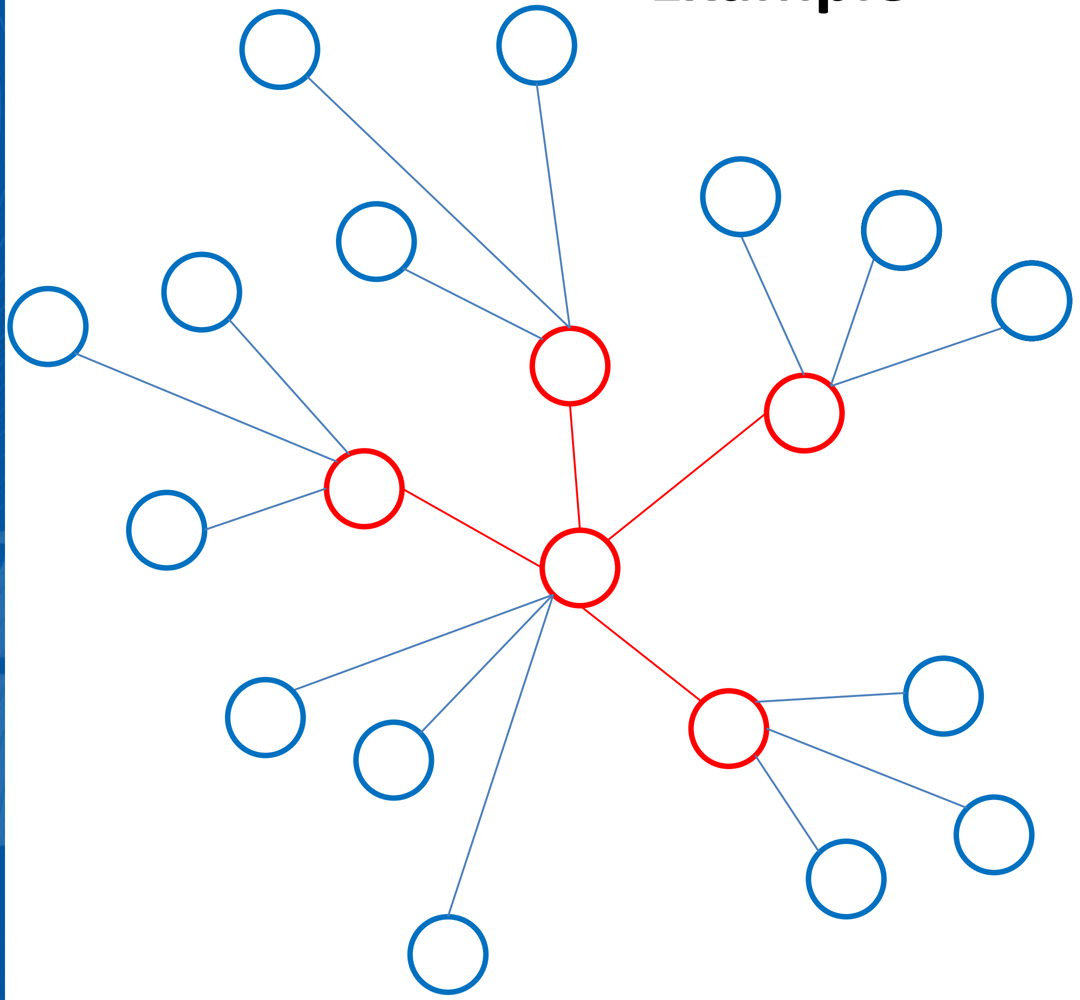
The nearest of each CoreNode is an AccessNode

Example



The nearest of each CoreNode is an AccessNode

Example

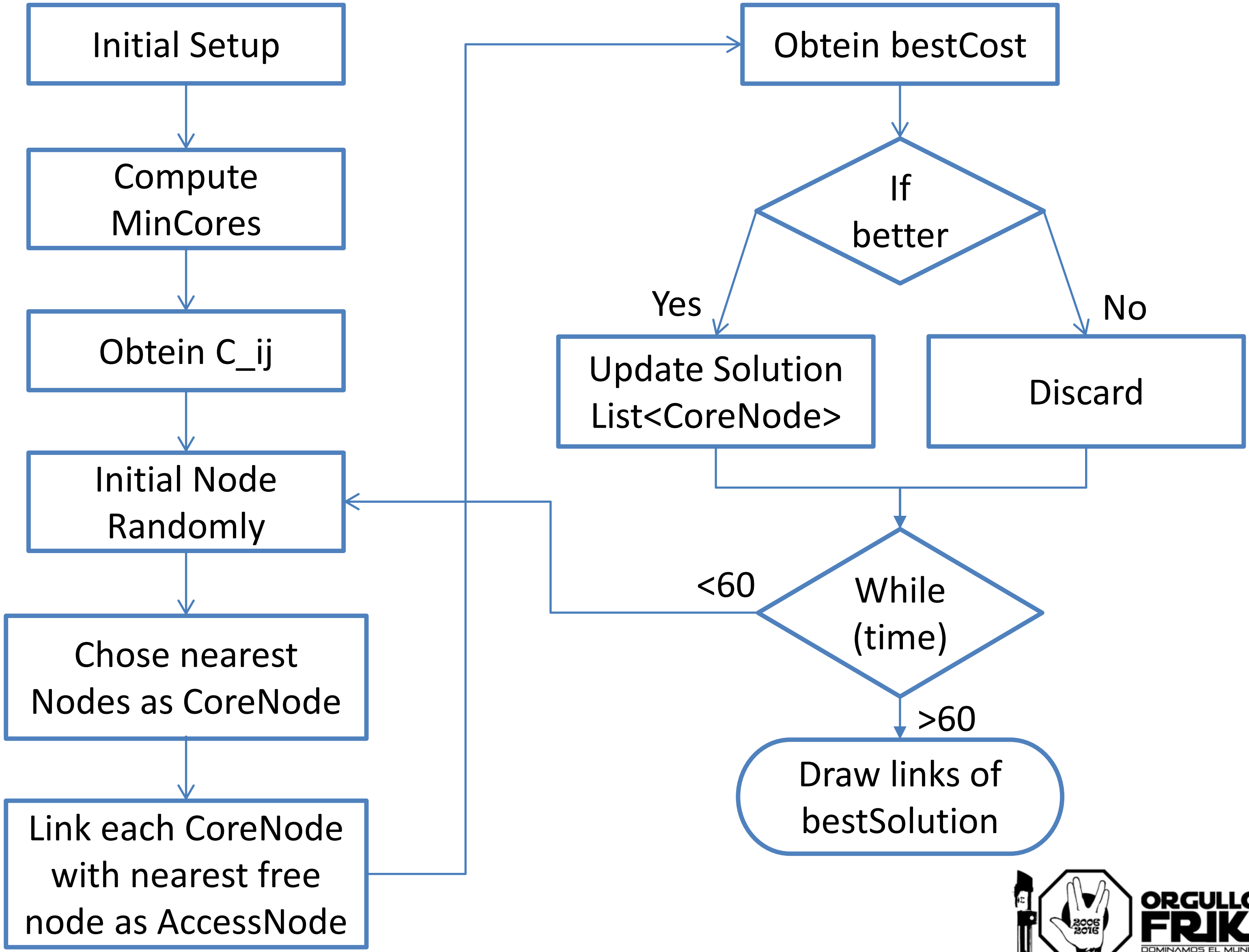


After the iterations each CoreNode have the same AccessNode

Algorithm

```
Calculate minCoreNodes related to maxNumAccessNodePerCoreNode and
totalNodes

First CoreNode chosen randomly

The nearest minCore are chosen

Foreach CoreNode in minCore{
        nearest maxNumAccessNodePerCoreNode  are chosen
}

Cost is stored as bestCost

While(time){
        Other First CoreNode choosen randomly
        The nearest minCore are chosen
        Foreach CoreNode in minCore{
                nearest maxNumAccessNodePerCoreNode  are chosen
        }
        If(cost<bestCost)
                bestCostUpdated
}
```

Flow chart



Initial Setup
Compute MinCores
Obtein C_ij
Initial Node Randomly
Chose nearest Nodes as CoreNode
Link each CoreNode with nearest free node as AccessNode
Obtein bestCost
If better
Yes
No
Update Solution List<CoreNode>
Discard
While (time)
<60
>60
Draw links of bestSolution

Objects are our friends

```
public class CoreNode{

int coreNode;
List<Integer> connectedNodes = new ArrayList<Integer>();

public CoreNode(){}

public int getCoreNode() {
    return coreNode;
}

public void setCoreNode(int coreNode) {
     this.coreNode = coreNode;
}

public List<Integer> getConnectedNodes() {
    return connectedNodes;
}

public void setConnectedNoder(List<Integer> connectedNodes) {
    this.connectedNodes = connectedNodes;
}

public void addConnectedNode(Integer node){
    this.connectedNodes.add(node);
}

public int getNumberOfConnectedNodes(){
    return this.connectedNodes.size();
 }
}
```

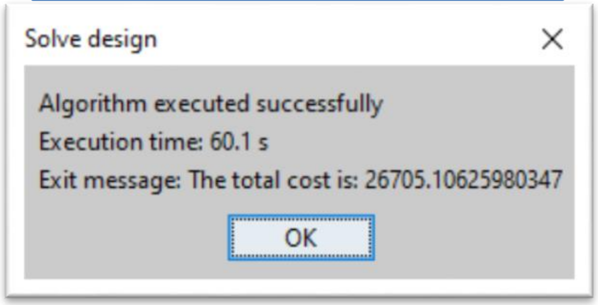
Attributes

Get y Set

Working methods

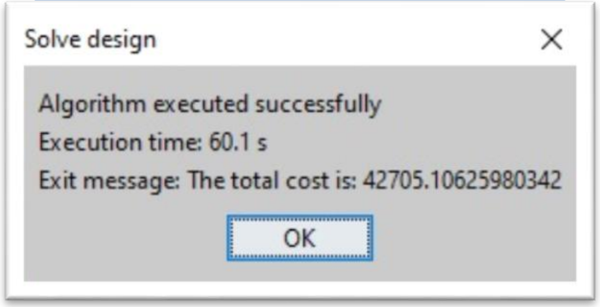
Solutions

CoreNodeCost = 100
MaxNumAccessNode = 5



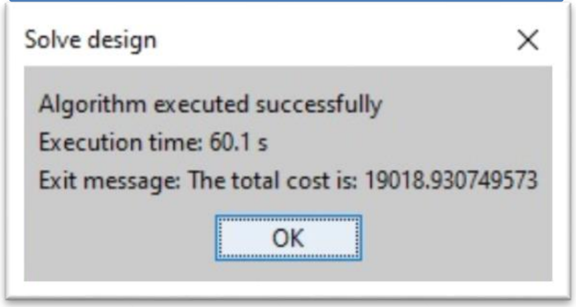
```
Algorithm executed successfully
Execution time: 60.1 s
Exit message: The total cost is: 26705.10625980347
```

CoreNodeCost = 500
MaxNumAccessNode = 5



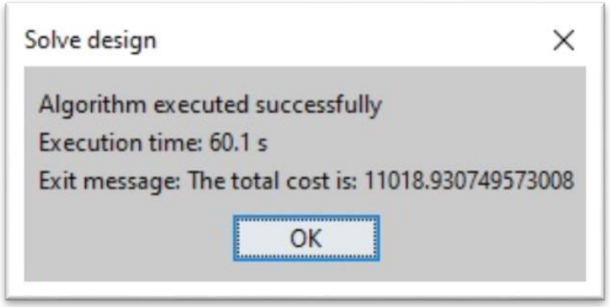
```
Algorithm executed successfully
Execution time: 60.1 s
Exit message: The total cost is: 42705.10625980342
```

CoreNodeCost = 500
MaxNumAccessNode = 10



```
Algorithm executed successfully
Execution time: 60.1 s
Exit message: The total cost is: 19018.930749573
```

CoreNodeCost = 500
MaxNumAccessNode = 10



```
Algorithm executed successfully
Execution time: 60.1 s
Exit message: The total cost is: 11018.930749573008
```

Universidad Politécnica de Cartagena

Bibliography:

- Optimization of Computer Networks – Modeling and Algorithms : A hands-On Approach.
- Notes and sketches from Pablo Pavón Mariño.
- Practice explanations from María Victoria Bueno Delgado and Juan Carlos J. Sánchez Aarnoutse.
- Api JavaDoc.

The end

AUTORES: Jairo Peña Iglesias y José A. López Pastor.